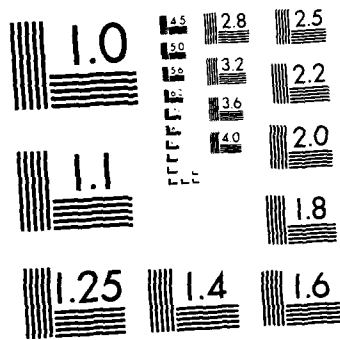


256

HL



DTIC FILE COPY

2

AFWAL-TR-87-2087



AD-A195 699

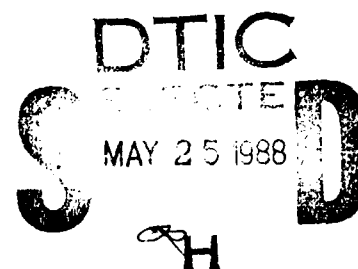
EMPIRICAL FLUTTER PREDICTION METHOD

Dr. J.K. Casey
GE Aircraft Engines
Advanced Technology Operation
Cincinnati, Ohio 45215

5 March 1988

Final Report for Period September 1984 - September 1987

Approved for Public Release; Distribution Unlimited



**AERO PROPULSION LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6563**

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Richard L. Martin

EDWARD L. MARTIN, Project Engineer
Propulsion Research Group

Francis R. Ostielek

FRANCIS R. OSTIELEK, Chief
Technology Branch

FOR THE COMMANDER

Robert L. Henderson

ROBERT L. HENDERSON
Deputy for Technology
Turbo Engine Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/POTX, W-IAFB, OH 45433 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY N/A		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) R87AEG		5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-87-2087	
6a. NAME OF PERFORMING ORGANIZATION GE Aircraft Engines	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Aero Propulsion Lab (AFWAL/POTX) AF Wright Aeronautical Labs	
6c. ADDRESS (City, State, and ZIP Code) GE Aircraft Engines 1 Neumann Way Cincinnati, Ohio 45215		7b. ADDRESS (City, State and ZIP Code) Wright Patterson AFB, Oh 45433-6563	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFWAL Aero Propulsion Lab	8b. OFFICE SYMBOL (If applicable) POTX	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract No. F33615-84-C-2457	
6c. ADDRESS (City, State and ZIP Code) Aeronautical Systems United States Air Force Wright-Patterson AFB, OH 45433		10. SOURCE OF FUNDING NOS.	
11. TITLE (Include Security Classification) Empirical Flutter Prediction Method		PROGRAM ELEMENT NO. 62203F	PROJECT NO. 3066
		TASK NO. 16	WORK UNIT NO. 17
12. PERSONAL AUTHOR(S) Dr. J.K. Casey			
13a. TYPE OF REPORT Final	13b. TIME COVERED FROM 9/84 TO 9/87	14. DATE OF REPORT (Yr., Mo., Day) 5 March 1988	15. PAGE COUNT 349
16. SUPPLEMENTARY NOTATION			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB. GR.	
		Flutter, Stability, Stall, Choke, Discriminant, Cluster	
19. ABSTRACT (Continue on reverse if necessary by block number)			
<p>Test points from the Annular Cascade Data Base, generated under Air Force Contract F33615-76-C-2035 have been analyzed, to predict from aeromechanical data which of 14 types of stability or instability would result. The basic approach was to identify for each pair of stability regions, linear combinations (hyperplanes) of the aeromechanical variables, whose numerical value would be above a critical level for all test points in one stability region and would be below the critical level for test points in the other stability region.</p> <p>It was found that 76% of the pairs of stability regions allowed a hyperplane to discriminate between the two regions, but for 24% a curved surface or nonlinear combination variables would be needed.</p> <p>Based on careful review of 85% of the 891 test points that were used to construct the hyperplanes, the hyperplanes correctly identify the stability condition of 59% of the</p>			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>		21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL R.J. Martin		22b. TELEPHONE NUMBER (Include Area Code) (513) 255-8210	22c. OFFICE SYMBOL AFWAL/POTX

points in a literal sense, but are correct in a broader practical sense for 79% of the points.

When the hyperplanes were applied to 51 validation test points taken from several actual engine/rig test data, they gave virtually no correct results. Perturbing the data $\pm 10\%$ brought no improvement.

This result is not immediately explainable. A review of the theory and application has offered no answer. However, several questions are raised by the review. Exploration of these questions might lead to an applicable empirical model for stability predictions.

Additional research is needed to determine if the hyperplanes predict the stability conditions of points from the Annular Cascade Data Base which were not used in their construction. Furthermore, it is not known how the operational map predicted by the hyperplanes compares with the true map. Work needs to be done to determine if there is a difference between the Annular Cascade and engines beyond what is reflected in the aeromechanical variables selected for analysis. There is the possibility that some engine points are surrounded by Annular Cascade test points having a different stability condition.



Accession For	
NTIS GRA&I	<input checked="checked" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

Summary

The design of advanced technology engines is often limited by compressor blade instability or flutter. The lack of adequate design tools can produce overly conservative designs with less than maximum levels of performance, or blade flutter can result, leading to a complete destruction of an engine. Consequently, the accurate prediction of the flutter boundaries is a key requirement for the successful design of future engines.

With the absence of adequate theoretical analyses, partially because of the lack of detailed flow information, but largely because of the complexity of the unsteady flow fields present during stall and choke flutter, it is necessary to develop valid empirical prediction design systems for flutter-free designs based on representative flutter data.

The available flutter data obtained on component or engine development programs were however very limited. These data provided only a small window in both operational characteristics and necessary data required to quantify detail aerodynamics and mechanical parameters in the flutter region. In view of these limitations, an Air Force sponsored program "Experimental Analysis of Blade Instability" (Contract F33615-76-C-2035) was initiated to widen the data window for both stall and choke flutter. In this program, numerous tests were conducted using a stationary annular cascade. A systematic variation of aerodynamic and structural parameters was made to provide a flutter data bank for both flutter regimes. The primary parameters varied included design reduced velocity, solidity, relative density, leading edge mach number, and incidence angle, all for front stage designs.

As an extension to this program, other variables such as camber, stagger, and frequency tuning were investigated, under Company sponsorship, to determine their effects on the stall and choke flutter boundaries. The data bank generated as a result of these experimental programs consists of several thousand data points.

An empirical prediction design system for the onset of flutter based on regression analysis, was attempted as part of the Annular Cascade program. The intent was to parameterize these data to identify the significant flutter parameters and subsequently to determine the appropriate design format, requirements, and procedures. The prediction correlation was not totally successful. Although the data did collapse using the developed Standard Day Flutter Parameter, FP_{SD} , the distinction between the flutter and stable points was at times ambiguous.

The results of this regression study indicated that either additional variables or other combinations of the present variables need to be established to provide adequate separation between the stable and flutter data points. Also the regression study showed the need to utilize geometric methodology that would directly address the geometric problem of separating the stable and flutter points, and the various types of flutter points.

The present program was undertaken to develop the appropriate geometric methodology and apply it to the Annular Cascade Data Base. The objective was to predict stall and choke flutter and then to validate the methodology using available engine data.

Acknowledgments

Drs. Ron B. Fost and Pat M. Niskode have provided all of the aeromechanical background and guidance which have gone into this work and, in particular, an in-depth knowledge of the Annular Cascade test vehicle and resulting data base. The Validation Points were supplied and interpreted by Dr. Fost.

Mike Stallone and, initially, Bob Jutras provided the initiative to conceive this work and the ongoing oversight.

Gerald Black provided expert programming help in converting GALACTIC from the Honeywell DPS90 to the VAX4.

Joe McKenzie, Vince Gallardo, and Dr. Frank Sagendorph provided valuable counsel.

TABLE OF CONTENTS

<u>SECTION</u>	<u>PAGE</u>
1.0 GALACTIC Computer Program	1
1.1 Programming Aspects	2
1.1.1 Program Input	3
1.1.2 Program Output	8
1.1.3 Logical Sequence	13
1.2 Mathematical Aspects	20
1.2.1 Matrix Geometric Analysis	20
1.2.2 Convex Hulls	31
1.2.3 Non-Objective Simplex Method	38
1.2.4 Discriminant Analysis	47
1.2.5 Relaxed Discriminant Method	72
1.2.6 Cluster Analysis	85

TABLE OF CONTENTS (CONTINUED)

<u>SECTION</u>	<u>PAGE</u>
2.0 GALACTIC Applied To Annular Cascade Data Base	94
2.1 Intersecting Pairs Of Stability Regions	96
2.2 Disjoint Pairs Of Stability Regions	99
2.3 Conclusions	100
3.0 EFAGHY Computer Program	101
3.1 Hyperplane File From GALACTIC	101
3.2 Test Point Files To Which Hyperplanes Are Applied	102
3.3 Input to EFAGHY	102
3.4 Voting By Hyperplanes	103
3.5 Comparison of Vote Count Efficiency	106
3.6 Mixed Voting Strategies	107
3.7 Summary Of Output Options	109
4.0 EFAGHY Applied To Annular Cascade Data Base	110
4.1 Detailed Examination Of 26 Points From File 55777B	110
4.2 Detailed Examination of 57 Points From File C2BDY	112
4.3 Generic Examination Of Data Base	113

TABLE OF CONTENTS (CONTINUED)

<u>SECTION</u>	<u>PAGE</u>
5.0 Selection Of Validation Points	115
5.1 Types of Test Points Proposed	115
5.2 Actual Test Points Selected	115
5.3 Aeromechanical Information Per Test Point	116
6.0 Application Of EFAGHY To Validation Points	118
7.0 Resolution Of Issues	132
7.1 Correctness Of GALACTIC Hyperplanes	132
7.2 Nonlinear Discriminating Surfaces	133
7.3 Additional Aeromechanical Variables	134
7.4 Data Base Selection	135
7.5 Validity Of The Validation Points	136
7.6 Annular Cascade Data Base Applicability	137
7.7 Implied Operational Map For Validation Points	138
7.8 Combination Stability Regions	138

TABLE OF CONTENTS (CONCLUDED)

<u>SECTION</u>	<u>PAGE</u>
8.0 Conclusions	140
8.1 Methodology For Analysis Of Large Data Bases	140
8.2 Hyperplane/Voting System Applied To Annular Cascade Data Base	141
8.3 Hyperplane/Voting System Applied To Validation Points	141
Appendix A - GALACTIC Listing	143
Appendix B - GALACTIC Hyperplanes	227
Appendix C - EFAGHY Listing	237
Appendix D - Sample GALACTIC Input	271
Appendix E - Sample GALACTIC Output	279
Appendix F - Sample EFAGHY Input	313
Appendix G - Sample EFAGHY Output	315

1.0 GALACTIC Computer Program

The name is an acronym for Geometric Analysis of Large Arrays
Containing Three or more Independent Coordinates.

As the name indicates, the program's capabilities are not at all specific to the flutter prediction problem while all its capabilities were intended to be useful for that problem, but are much more generic.

The fundamental point of view underlying GALACTIC is as follows: If, in the flutter prediction problem, there were only two (possibly three) independent variables - for example temperature and pressure, the test points would be plotted versus these two variables and labeled according to the stability condition that was observed. The points of the same stability condition would then be enclosed by a curve to define a region where that stability condition would be expected to prevail for engine data not in the data base.

The intent of GALACTIC is simply to extend to higher dimensional space the geometric analysis which is so intuitive and straightforward in two or possibly three dimensional spaces.

Some of the geometric tasks that the human eye does so effortlessly in two dimensions are as follows:

- o Normalize variables to comparable scales
- o Shape, size and orientation of groupings of points whether grouped by like stability condition or grouped by geometric adjacency
- o Extension of points of like stability condition into continuous enclosing regions where that stability condition prevails
- o Decision lines between the regions by which to predict the stability condition of a new point whose stability condition was unknown.

It is a simple yet accurate statement to say that GALACTIC and the methodology it embodies intends to translate these geometric tasks into algebraic tasks and then to extend these to higher dimensions.

The programming aspects will be described first, and then the underlying mathematical technology it embodies.

1.1 Programming Aspects

GALACTIC is written in FORTRAN 77. Its development was carried out mostly on the Honeywell DPS92 computer, but it was subsequently transferred to the VAX4 computer where it is now running. Its listing currently is 4875 lines long, of which 2661 lines are in the main program and the remaining 2214 lines are in subroutines.

Among the subroutines, twelve are from the Honeywell subroutine library supplied by the International Mathematical and Statistical Libraries; these routines were part of GALACTIC as it was being developed on the Honeywell computer and were subsequently transferred to the VAX4 when GALACTIC as a whole was moved to that computer. These routines are; ZX1LP, ZX3LP, LSVDB, LSVDF, LSVG1, LSVG2, UERSET, UERTST, UGET10, USPKD, VHS12, SROTG. They occupy 1142 lines of the listing.

The program can be run in batch or time sharing mode; it will detect which is being used. If in the batch mode, it must be supplied with a file containing input in the order in which it is needed.

The program input will be described first, then the output, followed by the logical sequence of the computation. The mathematical methodology is presented in the succeeding section. Actually, these four parts are so interdependent that they must be read in coordination with one another.

1.1.1 Program Input

The program input will be described in the time sharing mode with an occasional comment concerning batch usage. Generally, there is no distinction between the two; the program in one case reads the time sharing terminal and in the other case it reads a data file.

To facilitate the exact definition of the input, the read statement will be located in the program (listing as of 16:31 on July 20, 1987); for example, S580-2 means 2 lines prior to statement labeled 580.

All input read statements are preceded by prior prompts explaining what input is needed by the read statement. The prompts will be shown here, followed by any needed explanation. The statement number refers to the read statement, not the prompt.

S155+1 "Enter number and names of sample files, words/record, Y/N to clearfiles." Each record on an input file contains the data for a test point from the Annular Cascade Data Base. The names should be in single quotes for the VAX4. Clearfiles allow usage of temporary data files left over from a prior run. 'Y' or 'N' must also be in single quotes.

S250+1 "Enter N (LE 14), then name N variables." These identify the words from each test point record which will be used to describe a test point. The words are named by their numerical order.

S250+5 "Which other variable identifies subsets (0 if none)." This is the number of the word which contains the identification of the set to which the point belongs. If the set is unknown, the identification code "99" is used. The omission of a set identification variable, while possible in early versions of the program, may not lead to valid results in the current program, and should be avoided.

S290+1 "Enter N (LE 14), then ID of N sets to be excluded."

S295+4 "Enter N (LE 20), then name N points to be excluded."

S295+10 "Enter N, then N pairs of ID's to be labeled as the first."
 The test points from the data base are identified by the numerical order in which they are read in by the program, including test points read in from any prior files. If "a pair of ID's is labeled as the first," any points bearing the second ID will be altered so that they bear the first ID. Thus, the two sets are combined under the label of the first set.

S470+1 "Choose typical scale options (MIN, MAX, AVER, SIGMA, MID RANGE, SPECIAL." The typical value is subtracted from each test point variable, and the result is divided by the scale value. SIGMA means standard deviation of the sample. MID means average of the MAX and MIN values. The options listed, except for SPECIAL, call upon the program to use the values it has computed from the test points read in. There are six other options: MNU, MXU, AVU, SGU, MDU, RNU which are like the first six except that they were earlier computed from all the files in the sample (891 data points) from the Data Base and are contained in S110+4 to S120-2. SPECIAL allows the user to enter typical and scale values for the NV variables being used. In this case, there will be two additional prompts:

S540+1 "Enter typical values for the NV variables."

S570+1 "Enter scale values for the NV variables."

S580+3 "Enter count of bonded points (I & J forced into same cluster."

S600 "Enter the NBOND pairs of point numbers." The first read acquires NBOND, which is used in the second read. If point number I and point number J are found to be in different clusters, the two clusters will be combined.

S630+5 "Enter 0 or 1 to notwrite, or write following output options."

"For alldata: SMN, NTR, NPT, BAX, BTR, BPT."

S630+9 "For setdata: IPI, MEM, DSJ, HPL, BAX, PRJ, BTR, CGR, COR, EST, HPP, HPQ."

S630+13 "For clustrs: EDS, SDS, STP, FPT, MEM, CCD, BAX, PRJ, BTR, CGR, COR, SUR."

The input for the first read is a six-digit word, and for the second and third read, it is a twelve-digit word, the digits being 0 or 1. The specific output options will be described in the next section on Program Output. It is important to note that calculations are omitted within GALACTIC if no output derived therefrom is requested. Thus, the running time is heavily dependent on these three output options.

S3000+3 "Name Revised Sample File & Population Files (in sample format) (" " if none)."

The files read in originally at S155+1 were the original sample files. The Revised Sample file is intended to be used in the same way in subsequent runs of GALACTIC. It attempts to extract from the original sample files and the Population files the critical test points, that is the test points that would be needed in developing revised hyperplanes. The Population files are in the same format as the original and Revised Sample files and have test points in the same stability regions, but their points were not used in constructing the hyperplanes. The underlying thought is that if there are more test points available than can be analyzed simultaneously, it is possible to pick a sample, develop hyperplanes for the sample, test these against the larger population, and revise the sample to exclude the population points which were found to be redundant.

In this way, we hope that we can find a sample of critical points which will generate hyperplanes which will be satisfied by the entire population. This concept was developed due to memory restrictions on the Honeywell DPS92, but has not proven necessary on the VAX4 due to the latter's virtual memory system.

S3059+1 "Enter N (LE 14), then ID of N sets to be excluded."

S3059+4 "Enter N (LE 20), then name N population points to be excluded."

S3054+1 "Enter N, then N pairs of ID's to be labeled as the first."
This input has the same meaning as the input at S290+1, S295+4 and S295+10.

S3050+1 "Revised sample file, FILINR, already exists."

"Enter 1 to overwrite or 2 to enter new name."

"Enter name of revised sample file."

In case the output file FILINR already exists, the time sharing user may overwrite or supply a new name.

S3495+4 "Enter name of next population file (" " if none)."

S3491+2 "Enter N (LE 14), then ID of N sets to be excluded."

S3491+5 "Enter N (LE 20), then name N population points to be excluded." This input has same meaning as preceding input.

S4100+2 "Name input, output hyperplane file (" " to omit)." Enter, in single quotes, the name of the already existing file FILEHI of hyperplanes, as well as the designated name for FILEHO, the hyperplane file to be produced. Each record on these files describes a single hyperplane and consists of four more words than the number specified in S155+1. The first two words contain the identification for the two sets, the second two words contain the constant term of the hyperplane equation for each of the two sets and the remaining words are coefficients for each word in the test point records. In case FILEHO is an already existing file, the usual overwrite or name revision is provided for.

- S4149 "Choose to save old, new planes (O, N) for sets ___, ___."
The user specified Old or New. This choice is made on the basis of output available earlier in the current run, as compared with output available in the prior runs that produced the old hyperplane equation.
- 9010+3 Enter O, N netto, to transform data per cluster N axes, N LE O per all data."
- 9020+1 "Enter max number of axes to be used."
This causes the data originally read in accord with S155+1 to be read again and then transformed from the original variables into new variables defined as the leading axes of one of the clusters, or of the entirety of the input data, i.e. the alldata axes. This option requires that the test points be similarly transformed any time they are read afresh from the Data Base files. Since this is not currently done, the option is safe to use only if the calculations required by the user do not entail rereading the test point data. This, for example, is the case if only cluster analysis is required. (This restriction on the use of this option could be easily removed if the need justified doing so.)
- S9140+2 "Enter O, 1 ifnot, if above cluster CG's to be treated as points." This option occurs only if the prior option was not exercised. It is subject to the same restriction. The idea of the option is to condense the clusters into single points situated at their center of gravity, and to reanalyze the problem in this simplified form, finding, for example, new clusters (or galaxies) made up of the old clusters - much in the spirit of classical mechanics. This is, therefore, allowed to remain despite the restriction on its use, as a vestige of a possible future activation.
- S9170+1 "Enter 0,1,2 as there arent, are more cases, change case."
If no more cases, the program stops at S9180. If there are more cases, the program either goes to the beginning S130 or to the input on bonding and output codes S580 saving the user the burden of repeating all the input.

1.1.2 Program Output

The major output is the updated set of pairs of hyperplanes, which is written out on a data file. The format consists of the stability code for the first set, the stability code for the second set, the left-hand side constant for the first set, the left-hand side constant for the second set, followed by a coefficient for each of the words in the format of the test points supplied from the Angular Cascade Data Base.

Next we should note that the program input is printed out when the program operates in batch mode, so that the hard copy output would include prompts and responses in very much the same way as if the program were run in time sharing mode.

The following description of output will consist in an explanation of the thirty 0 or 1 digits in the three output code words that control program printout. Each one of these digits has a three-letter abbreviated description intended as a memory aid to the user. Digits are counted from the left of the code word; 1 calls for printout and 0 suppresses it. The first code word IOUTPUT refers to preliminary calculations and has six digits. The second code word IOUTPS refers to the set discrimination calculations and has twelve digits. The third code word IOUTPC refers to cluster analysis and has twelve digits also.

Output from Preliminary Calculations

- | | |
|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SMN | (Digit 1 of IOUTPUT) Prints out for each set, the set number, the set code and the number of points. Also prints a summary of each variable showing minimum, midvalue, maximum, range, average, and sample standard deviation. |
| NTR | (Digit 2 of IOUTPUT) Prints the transformation from the original variables to the variables normalized by a typical value and scale factor. Also, the reverse transformation. |
| NPT | (Digit 3 of IOUTPUT) Prints the data points normalized by a typical value and a scale factor. |

- BAX (Digit 4 of IOUTPUT) Prints for the entirety of the test points: the center of gravity, the semiaxes of bottle containing the data, the length of the semiaxes.
- BTR (Digit 5 of IOUTPUT) Prints transformation to and from the axes for the entirety of the test points.
- BPT (Digit 6 of IOUTPUT) Prints data points transformed to the axes for the entirety of the test points.

Output Specific To Set Discrimination

- EST (Digit 10 of IOUTPS) Prints out a matrix of Euclidean distance between pairs of sets, that is, the minimum Euclidean distance between pairs of points, one in one set, one in the other set. Also prints the maximum distance between points in the same set. In addition, the points providing these distances are shown (both if between sets, one if within sets).
- IPT (Digit 1 of IOUTPS) Prints out points of each set which are vertices due to their having a variable with a maximum or minimum value. Also, the points found to be redundant are expressed as convex linear combinations of the vertices.
- MEM (Digit 2 of IOUTPS) Prints for each point the number of the set to which it belongs. A minus sign is used to indicate that the point has been found to be redundant.
- DSJ (Digit 3 of IOUTPS) Prints whether the convex hulls of pairs of sets are disjoint. Disjointness is sought using only the first alldata axis, then the first two, etc., using no more than are necessary. Each step is reported on. If the pair of sets is not disjoint, then an intersection point is exhibited as a convex linear combination of the vertices of each set.
- HPL (Digit 4 of IOUTPS) Prints out hyperplanes to discriminate between each pair of sets. The hyperplanes are developed using as few axes as possible, beginning with the number of axes found necessary above (see DSJ). If the sets were previously judged to be disjoint, then the hyperplanes are sought with a gap; otherwise, the overlap option is used. If hyperplanes cannot be gotten successfully, then the opposite case (overlap or gap) is tried and the result is accepted. The success at each step of

this process is reported. When this is completed for all pairs of sets, the distance to each test point along the normal to each pair of hyperplanes is computed and marked with E in case the point violates a hyperplane. This is printed as well as a tally of the E's. The planes may be adjusted to remove some small violations and this too is reported.

HPP (Digit 11 of IOUTPS) Prints for each test point on a Population file, the distance to the point along the normal to each pair of hyperplanes, with , , or = to indicate that the point is beyond both planes, falls short of both planes, or is between both planes. Once this is completed a tally is printed showing the number of points on the Revised Sample File, the number of these which were from the original or input sample files and non-redundant, the balance being from the Population files with identification code not in the input sample file or misclassified by hyperplanes. Among those misclassified by the hyperplanes are various subcategories. The subcategories indicate whether by adjusting gaps between the planes the violation could be removed and, if so, whether not only the correct stability set but also an incorrect stability set might be indicated. The same subcategories can apply when only incorrect stability sets are indicated. (These types of category are examined more completely in the EFAGHY program.)

Non-Optional Output if computed: Prints hyperplanes in terms of the original input variables; the prior printout showed the hyperplanes in terms of the original variables transformed by subtraction of a typical value, divided by a scale value, then with center of gravity subtracted and converted to all data axes. The conversion of the hyperplanes to original variables facilitates their application to a large number of points expressed in terms of the original variables.

HPO (Digit 12 of IOUTPS) Prints for each test point on original sample files, the distance to the point along the normal to each pair of hyperplanes. This is calculated using the test points in the original variables and the hyperplanes in the original variables. Thus, this output is a check on prior output, HPL, which it should duplicate.

Output Specific To Cluster Analysis

EDS (Digit 1 of IOUTPC) Prints matrix of Euclidean distances between each pair of normalized points.

SDS (Digit 2 of IOUTPC) Prints matrix of Stepping Stone distance between each pair of normalized points.

STP (Digit 3 of IOUTPC) Prints matrix of count of steps between each pair of normalized points.

FPT (Digit 4 of IOUTPC) Prints matrix of frontier points in going from one point to any other point. The ij entry is the frontier point encountered first in going from point i to point j .

MEM (Digit 5 of IOUTPC) Prints for each test point the number of the cluster to which the point belongs, with a minus sign to denote an interior point. Also prints the count of set members in each cluster.

CCD (Digit 6 of IOUTPC) Prints the Stepping Stone distance between clusters as well as the Stepping Stone diameter of each cluster, that is, the maximum Stepping Stone distance between pairs of points which both belong to the cluster.

SUR (Digit 12 of IOUTPC) Prints out the fit of a quadric surface to a cluster. This option should not be exercised since the validity of the output is questionable. Clarifying this matter has not been done since it has not had high priority and because the shape and size of a cluster is adequately described by the calculation of cluster axes. The present option has, however, been retained, since it is worth validating.

Output Common To Geometric Analysis of Both Sets and Clusters
(The generic term "group" will be used for "set" or "cluster".)

- BAX (Digit 5 of IOUTPS, Digit 7 of IOUTPC) Prints semiaxes of each group, and the length of each semiaxis.
- PRJ (Digit 6 of IOUTPS, Digit 8 of IOUTPC) Plots for each group the projection of its points on each axis. If the points are represented with coordinates on each axis, the projection is simply the plot of the coordinates for each axis. This provides some visual grasp of how the points are concentrated in the group. The plots are printer plots and quite adequate for their purpose.
- BTR (Digit 7 of IOUTPS, Digit 9 of IOUTPC) Prints transformation to and from bottle axes.
- CGR (Digit 8 of IOUTPS, Digit 10 of IOUTPC) Prints center of gravity of each group. If the group is a cluster, then there is printout as to where the center of gravity is located relative to its cluster.
- COR (Digit 9 of IOUTPS, Digit 11 of IOUTPC) Prints correlation matrix for each group. Also prints the volume of the points in the group as reflected by the determinant, labeled the CVOLUME, as well as the volume of the bottle containing the points. This output is not valid, but its correction has been deferred in favor of higher priority items. The option has not been suppressed since it is conceptually worthwhile and deserves completion.

1.1.3 Logical Sequence

The logical flow in GALACTIC is extremely simple, going directly from beginning to end with no major loops; the only deviation is a section of code in the form of an open subroutine which is executed once for sets from early in the code and is executed again for clusters in the sequence in which it is located.

Accordingly, the artificial format of block diagrams will not be needed. Each section of code will be denoted using statement numbers, since the VAX computers do not use labels for lines. The beginning or end of a section of code will be denoted for example as S580-2 meaning the second line preceding statement labelled 580. The statements are labelled sequentially with few exceptions.

Preliminary

beginning to S110+1	Data Organization
S110+3	Detect whether batch or time sharing mode
S110+4 to S120-1	Summary data for 891 Annular Cascade Test Points, 16 variables, min, max, average, std dev, mid point, range
S120 to S210-1	Print title, date Read names of data files from Annular Cascade Data Base Open temporary data files: 10, 11, 12, 13, 14, 15, 22, 23, 24, 25
S210 to S300-11	Read which variables to be included, any sets or points to be excluded
S300-10 to S360+2	Read data files 01 into XINF tables, named variables only Accumulate in VARMOM min, max, sum, sum of squares for each variable Count sets in ISETNAM, points/set in NPINSET Set id per point in IPINSET
S360+3 to S450-1	Print points/set and set id Change id to set number in IPINSET
S450 to S580	Choose option for typical, scale value for each variable
S580+1 to S630-1	Read which clusters are bonded, NBOND, IBOND
S630 to 6640-1	Read output codes into IOUTPUT, IOUTPS, IOUTPC If required, go back to do S495-S510+2 to print summary data for each variable
S640 to S740+1	Transform XINF data using typical and scale values, print, save new XINF on file 10
S750 to S809-1	Calculate EDIST matrix of Euclidean distance between pairs of points Save on files 11 and 12 Initialize file 13 to matrix of 1's with 0 on diagonal Point min distance between sets, max distance between points and within sets
S809 to S812+1	Call VERTEX to decide which points have a max or min coordinate

S812 to S790+1 Set parameters for open subroutine (S7075 to S7610+5)
to do basic geometric analysis by set
Initialize "group" to mean "set"

Basic Geometric Analysis (open subroutine; "group" = "sets" or "clusters")

S7075 to S7090+1 Initialization

S7090 to S7130 Read data from file 10 into XINF, rearranging by group

S7130+1 to S7160 Find CG for each group, subtract from data in XINF

S7160+1 to S7230-1 Call BOTTLE to find direction and length of axes
Count of non zero axes into NAINCL
Print diagnostic if dot product of axes exceeds 10^{-5}

S7230 to S7240-1 Projection of points in group on each axis

S7240 to S7320 Transformation to and from axes

S7330 to S7340 Go back to 7130+1 for next group, if any

S7340+1 to S7350+1 Print out CG of each group

S7360 If "group" means "set," skip to S465

S7360+1 to S7460+1 Read in test points from file 01, adjust for typical
and scale values
Determine if CG within, outside, toward a cluster

S7465 to S7550 Compute correlation matrix VAR for points represented
by coordinates on bottle axes

S7550+1 to S7610 Compute determinant of VAR matrix, and CVOLUME of group
Compute BVOLUME, i.e. volume of box with same axes

S7610+1 to S7610+6 If "group" means "set," go to S970
If "group" means "cluster," go to S7615

Discriminant Analysis

S970 to S990-1 Use only axes at least 5% as long as first axis
Call VERTX to find vertices for each set
Restore XINF from file 10

S990 to S9105-1 Initialize axes, axis length and set CG to 0
Subtract CG from XINF data
Call BOTTLE for all data axes (i.e. irrespective of set)
Print axes, length, cg, transformation to and from
all data axes, test points in terms of all data axes

S990 to S9105-1	Initialize axes, axis length and set CG to 0 Subtract CG from XINF data Call BOTTLE for alldata axes (i.e. irrespective of set) Print axes, length, cg, transformation to and from alldata axes, test points in terms of alldata axes
S930 to S2080+1	Sort both sets of vertices, combine
S2085 to S2246-1	Determine if points not yet found to be vertices are really vertices or are redundant Exhibit redundant points as linear convex combination of vertices
S2246 to S2250+1	Print for each point its set number, with minus for redundancies
S2260 to S2340	Call INSECT to determine discriminant feasibility (first method), that is if sets are disjoint. Begin with 1st alldata axis, including successive axes until disjointness occurs or axes exhausted. If not disjoint, using all axes, exhibit a point as convex linear combination of points in each set. Do this for all pairs of sets.
S2370 to S2390+2	Copy XINF in alldata axes into file 22
S2400 to S2447+2	Call DISCRM to determine pair of hyperplanes by Relaxed Discriminant method (3rd method) Begin with the number of alldata axes found by INSECT to suffice for disjointness If DISCRM finds these do not suffice, more axes are included If INSECT found sets are disjoint, DISCRM begins with gap case If INSECT found sets are not disjoint, DISCRM begins with overlap case If DISCRM fails, it goes to opposite case using all axes and accepts answer
S2447+3 to S2513+1	Read test points in alldata axes from file 22 Calculate for each point its distance orthogonal to each pair of hyperplanes Mark with E or B if stability condition of point is known and point is on wrong side of a pair of planes or in band
S2517 to S2515+1	Adjust hyperplane to eliminate borderline vioiations Hyperplane orientation not changed

S2520 to S2510+1	Print set number for each test point, minus if redundant If point is far from planes, on correct side, it is redundant
S3000 to S3064-1	Open Revised Sample file FILINR as file 03 Read name of Population file FILINT Read id of sets or points to be excluded or sets to be combined Invent new name if FILINR already exists
S3064 to S3080-1	Skip if FILINR is blank Read test points from 01, copy onto 03 unless excluded or redundant
S3080	Set ISETNAMP equal ISETNAM
S3080+1 to S3320	Skip to S3499 if FILINT is blank Open FILINT as file 02 Read point from FILINT If point is not excluded and has a set not treated in preceding discriminant analysis, copy it onto FILINR Convert point to alldata axes
S3320+1 to S3470+1	Apply hyperplane equations to point Determine for each pair if error, if distance to point along orthogonal to planes is > , < or between (=) to constants positioning the pair of planes
S3470+2 to S3494+1	If set membership of point is known and a hyperplane is violated, write point onto file 03, tallying type of error Print distance to point in direction orthogonal to planes with mark > , < , = Do for all points on FILINT, file 02
S3495 to S3499-1	Read in name of next population file with new exclusions possible Redocalculations, going back to S3100
S3499	If no more population files, FILINT, rewind 03
S3499 to S3498-1	Print out E totals, if any
S3498 to S3493+2	Print tally of types of error, cases of false identification, number and source of points on revised sample file 03

S4000 to S4060	Transform hyperplane equations to original input variables
S4060+1 to S4067+2	Apply transformed hyperplanes to test points on file 01, compute distance to point, orthogonal to plane as check against same calculation in alldata axes
S4067+3 to S4121-6	Read in names FILEHI and FILEHO of old and new hyperplane files Open these files as 04 and 07, create new name if necessary for FILEHO
S4121-5 to S4200+5	Read in FILEHI Copy hyperplane from FILEHI onto FILEHO if no new hyperplane was developed for the same pair of sets, or if old hyperplanes same as new Otherwise ask user if in time sharing; use old if in batch Rewind files and close

Cluster Analysis

S6000 to S6030-1	Initialization
S6030 to S6200	Calculate SDIST matrix of Stepping Stone distance between each pair of points, IDIST matrix of number of steps between pairs of points and IFRONT matrix of pair of frontier points in going from one point to any other point This makes one sweep through the data Number of changes are counted in ICHANGE2, number of sweeps is counted in ICHANGE1
S6200+1	If ICHANGE2 is 0, go to S6220
S6200+2 to S6210-1	Up ICHANGE1 by 1 Print out ICHANGE2 and computer cost for sweep User decides whether to continue; if yes, go to S6030
S6210 to S6360+1	Delete temporary files no longer needed (12, 13, 14, 15) or (22, 23, 24, 25) Print SDIST, IDIST, IFRONT
S6370-7 to S6550+1	Identify clusters ICLUSTER has point numbers (columns) for each cluster (row)

S6550+2 to S6730+1	Find subclusters
S6730+2 to S6880+3	Combine clusters that are bonded Combine if cluster has only one point
S6890 to S6950	Calculate and print IPINCL, showing for each point the number of its cluster
S6950+1 to S7020	Count points in each cluster, by set
S7020+1 to S7060	Find Stepping Stone distance between clusters Also max Stepping Stone distance within clusters
S7061 to S7075-1	Initialize "group" to mean "cluster" for Basic Geometric Analysis open subroutine Execute open subroutine
S7615 to S7720	Quadratic equation enclosure for each cluster

Termination Procedure

S9000-2 to S9010	Print computer time used
S9010+1 to S9130+1	If required by user, transform data to alldata axes or axes of some cluster and go to S450
S9140 to S9170-1	If required by user, condense clusters to their CG and go to S580
S9170 to S9180+1	User decides whether to stop, to go to S130 to begin a new case, or to go to S580 to redo the same case but with possible changes in output or bonding between clusters

1.2 Mathematical Aspects

1.2.1 Matrix Geometric Analysis

In this section several basic geometric questions will be addressed, such as the shape and size of a set of points, the coherence between points and the correlation between variables. This may be termed matrix geometric analysis since the geometric observations are simply interpretations of standard matrix analysis.

The number of test points will be denoted by m and the number of variables per test point will be denoted by n . The data points will be written in an $n \times m$ matrix A . Here m may be much bigger than n . Also the number of linearly independent rows in A may be less than either m or n .

Normalization of Data Points

In treating test data as geometric points it is helpful to normalize coordinates to have the same range. For example a mach number reading may be 0.9 while a temperature reading may be 900.

To normalize a variable, it is customary to subtract a typical value and then multiply by a scale factor. The typical value might be any measure of central tendency, like the mean, the mid point of the range, etc. The scale factor might be the reciprocal of a measure of dispersion like the standard deviation, the range, etc.

The center of gravity of the set of points is the point each of whose coordinates is the average of that coordinate of the points of the set.

Shape of a Set of Points Using Eigenanalysis

A basic geometric question is what sort of shape is formed by these points, what sort of shape would enclose them.

One way to answer this question is to consider all vectors $y = A^T x$ where x has unit length $x^T x = 1$. The possible y 's consist of all possible linear combinations of the points in A , including in particular the points themselves. The question of which y has maximum length is answered by choosing x so as to maximize $y^T y$ subject to the constraint $x^T x = 1$. This written with a Lagrangian multiplier s requires maximization of

$$x^T A A^T x - s(x^T x - 1)$$

which occurs when

$$A A^T x - s x = 0$$

Thus the x that minimizes y is the eigenvector of $A A^T$ with greatest eigenvalue s and the corresponding value of $y^T y$ is s .

The successive eigenvectors and eigenvalues give the same answer but in directions orthogonal to the prior y 's. If x^1 and x^2 are distinct eigenvectors

$$x^{2T} A A^T x^1 = 0$$

and so the y 's are also orthogonal

$$y^{2T} y^1 = 0$$

The y 's thus generated form a set of orthogonal vectors of lengths given by the square root of the corresponding eigenvalues s .

The y vectors thus gotten are themselves eigenvectors of $A^T A$ since if $y = A^T x$ and x is an eigenvector, then

$$A^T A y = A^T A A^T x = s A^T x = s y$$

The ellipse satisfied by the y eigenvectors when they have the length $(1/s)^{1/2}$ is given by the quadratic equation

$$1 = y^T (A^T A)^{-1} y$$

If y is an eigenvector of unit length and y is substituted into the quadratic equation the right hand side is r^2/s so that r must be \sqrt{s} for the vector to satisfy the equation of the ellipsoid. This says that the ellipsoid contains the y axes as required.

For example if $m = 3$ and $n = 2$, consider points $(1, 0)$, $(0, 1)$, $(1/2, 1/2)$. Then

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1/2 & 1/2 \end{bmatrix} \quad AA^T = \begin{bmatrix} 1 & 0 & 1/2 \\ 0 & 1 & 1/2 \\ 1/2 & 1/2 & 1/2 \end{bmatrix}$$

The eigenvectors and eigenvalues of AA^T are

$$\begin{array}{lll} 1/\sqrt{3} & , & 1/\sqrt{3} & , & 1/\sqrt{3} & \text{with eigenvalue } 3/2 \\ 1/\sqrt{2} & , & 1/\sqrt{2} & , & 0 & 1 \\ 1/\sqrt{6} & , & 1/\sqrt{6} & , & -2/\sqrt{6} & 0 \end{array}$$

Since m exceeds n , the $m \times m$ matrix AA^T is necessarily singular, being of rank at most n .

The corresponding $y = A^T x$ vectors are, respectively

$$\begin{array}{lll} \sqrt{3}/2 & , & \sqrt{3}/2 & \text{which is of length } \sqrt{3/2} \\ \sqrt{2}/2 & , & \sqrt{2}/2 & 1 \\ 0 & , & 0 & 0 \end{array}$$

which are eigenvectors of $A^T A = 1/4 \begin{pmatrix} 5 & 1 \\ 1 & 5 \end{pmatrix}$ with eigenvalues $3/2$ and 1 .

The ellipse with these axes is $1 = y^T (A^T A)^{-1} y$

$$5 y_1^2 + 5 y_2^2 - 2 y_1 y_2 = 6$$

The ellipsoid thus gotten is representative of the shape of the set of points, especially if the original points have been normalized so that each coordinate has a commensurate range, and has been shifted to a central value.

The above development, drawn from classical eigenanalysis provides an ellipsoid which has the shape of the array of data points. When m is a large number, computation of the eigenvalues x can be an imposing task. This is less so for the eigenvalues y since the $A^T A$ matrix is $n \times n$ and much smaller than the AA^T matrix. Nonetheless eigenanalysis can be difficult.

It is therefore useful to observe that the problem when seen from the viewpoint of matrix norms can be considerably simplified.

The problem of finding the y eigenvectors may be stated as: to find y such that $\|Ay\|$ is maximized while $\|y\|$ is kept constant. Here the vector lengths are stated in the usual Euclidean norm:

$$\|y\| = [y_1^2 + y_2^2 + \dots + y_n^2]^{1/2}$$

This approach will be presented next.

Ellipsoid as Container

While the ellipsoid as gotten above should reflect the shape of the points represented in A , and contain the axes, it is not clear that all points are necessarily within the ellipsoid. Consider the rectangular box whose axes are those of the ellipsoid.

Any point may be represented as a combination of eigenvectors y^i of unit length:

$$y = r_1 y^1 + r_2 y^2 + \dots$$

Suppose that a point say y^0 of A were outside the box then r_i^2 exceeds s_i for some i and y^0 , not y^i the actual eigenvectors

would have given the maximum of $y^T A A^T y$ for components of points orthogonal to the prior eigenvectors. But this is contrary to assumption that y^i is an eigenvector; it follows that all the points of A are within the rectangular box.

However it is not clear that there may not be points lying inside the box but outside the ellipsoid. Again consider a point y^0 not along an eigenvector to be represented in terms of the eigenvectors y^i of unit length. For this point not to be an eigenvector it is necessary that when x^0 is chosen (so that $y^0 = A^T x^0$) all components are zero but the one corresponding to this point, the result $y^T y$ is less than s_1 :

$$r_1^2 + r_2^2 + r_3^2 + \dots < s_1$$

Similarly when the same is done for the components of y orthogonal to prior y^i

$$\begin{aligned} r_2^2 + r_3^2 + \dots &< s_2 \\ r_3^2 + \dots &< s_3 \\ \text{etc.} \end{aligned}$$

When the point y is substituted into the equation of the ellipsoid the result is

$$\frac{r_1^2}{s_1} + \frac{r_2^2}{s_2} + \frac{r_3^2}{s_3} + \dots$$

and the question is how this compares with 1.

The proceeding inequalities may be summed: $1/s_1$ times the first, $(1/s_2 - 1/s_1)$ times the second, $(1/s_3 - 1/s_2)$ times the third, etc. The result is

$$\frac{r_1^2}{s_1} + \frac{r_2^2}{s_2} + \frac{r_3^2}{s_3} + \dots < n - \frac{s_2}{s_1} - \frac{s_3}{s_2} - \dots - \frac{s_n}{s_{n-1}}$$

which lies between 1 and n . The right-hand side can be shown to be less than

$$n - (n-1) \left(\frac{s_n}{s_1} \right)^{\frac{1}{n-1}}$$

The ratio of the maximum to minimum eigenvalues is one of the measures of the condition of a matrix.

This implies that if the ellipsoid is enlarged so that the unity on the right hand side of its equation is replaced by $n - \sum_{i=1}^{s_{i+1}} s_i$ then it will enclose the data points. This however may be a less efficient container for the points than the box having the same axes as the ellipsoid.

Shape of a Set of Points using L_1 Eigenanalysis

An alternate approach is to seek a direction y in which the rows of A have the greatest projection vector. The projection components are the entries in $p = A_y$. The projection vector is greatest in the sense that the maximum projection component is greatest for all y 's of unit length. The maximum of a vector is a norm, called the L_1 norm so that

$$\|p\|_1 = \max \{p_1, p_2, \dots, p_m\}$$

The best y solves the problem:

$$\text{maximize } \{ \|A_y\|_1 - s(\|y\|_2 - 1) \}$$

where the usual Euclidean norm constrains y to have unit length. (Note that the Euclidean or L_2 norm is the usual vector norm; it is implied when the type of norm is not designated.) Whatever y is found, $\|A_y\|_1$ has the form $a_I^T y$ where a_I is some row of A . Differentiating with respect to the components of y gives

$$a_I - 2sy = 0$$

or

$$y = a_{I.} / \|a_{I.}\|_2$$

upon imposing the constraint. Thus y is in the direction of some row of A . Consider the projection with row J

$$a_{J.}^T y = a_{J.}^T a_{I.} / \|a_{I.}\| = \|a_{J.}\| \cos(a_{I.}, a_{J.}) \leq \|a_{J.}\|$$

Thus the projection is greatest if y is in the direction of the row of A having greatest length.

Having found the first direction y^1 in this way, other directions are sought, each orthogonal to the prior y 's. These can be gotten by the same approach, provided the matrix A is altered so as to remove the projections in the direction of the prior y 's.

Thus if $a_{J.}^{(k)}$ was the latest J th row of A and $a_{I.}^{(k)} / \|a_{I.}^{(k)}\|$ was the latest direction y then the new row would be

$$a_{J.}^{(k+1)} = a_{J.}^{(k)} - \frac{a_{J.}^{(k)} \cdot a_{I.}^{(k)}}{\|a_{I.}^{(k)}\|_2^2} a_{I.}^{(k)}$$

Note that the problem: find y to maximize

$$\|A y\|_1 = s (\|y\|_2 = 1)$$

is the eigenvalue problem described previously except that there the Euclidean, or L_2 , norm $\|Ay\|_2$ was used, while here the L_1 norm is used.

Axes as New Variables

The axes gotten by L_2 or L_1 eigenanalysis provide a new orthogonal coordinate system which is more natural for the data points than the original coordinate system. This is especially true if the axes are quite short in some direction, indicating a combined variable which is nearly constant for the points in question. If these points are a subset, then it suggests that the subset may be distinguished best from other subsets by the variable which is nearly constant for the subset.

The Projection, Variance and Correlation Matrices

The matrices AA^T and A^TA have distinct and useful interpretations in themselves, apart from their usefulness for other purposes.

The matrix AA^T is an $m \times m$ matrix, where m is the number of data points. Its i, j entry is

$$\|a_i\| \|a_j\| \cos(a_i, a_j)$$

that is the dot product of the i th data point and the j th data point, or equivalently the projection of one on the other. Thus the diagonal entries of AA^T contain the Euclidean length of the data points. The matrix AA^T is therefore the projection matrix.

If the rows and columns of AA^T are divided by the square root of their diagonal entries, the resulting matrix is composed of the cosine of the angle between pairs of points. To the extent that cosines are near one, the data points are confined to a narrow cone.

The A^TA matrix is an $n \times n$ matrix, where n is the dimension of the space, that is the number of variables that constitute the data point. Its i, j entry is

$$a_{1i}a_{1j} + a_{2i}a_{2j} + \dots + a_{ni}a_{nj}$$

This provides the covariance of the readings of the i th and j th variables according to the formula

$$\text{cov}(a_i, a_j) = \frac{1}{n} \sum_{k=1}^n a_{ki}a_{kj} - \frac{1}{n} \sum_{k=1}^n a_{ki} \sum_{k=1}^n a_{kj}$$

The last term simply normalizes A by subtracting from each entry, the mean of its column. If this is done first then the i, j entry of the resulting A^TA matrix is n times the covariance of the i and j normalized data reading.

Thus $A^T A$ with the indicated normalization is the covariance matrix of the data.

The diagonal entries of the covariance matrix are the variances, that is the standard deviations of the data, squared.

The covariance matrix becomes the correlation matrix if each row and column is divided by the square root of their diagonal entries. The ij entry in the correlation matrix is then the correlation between the readings of the i th and j th variables.

The off diagonal entries lie between +1 and -1 and indicate the degree to which two variables change together (+) or oppositely (-). The main diagonal is of course all ones. To the extent that two variables are correlated, one variable is superfluous.

Volume of a Set of Points

Another useful piece of geometric information is the volume of the data. In three dimensional space the volume of the parallelepiped having three linearly independent vectors as edges, is given by the determinant of the 3×3 matrix having the coordinates of the i th point in the i th row. Similarly for two dimensions.

For n points in n dimensions, the volume of the parallelepiped P is related to the volume of the n dimensional unit cube C by

$$\int_P dV = \int_C J dV' = J$$

where J is the Jacobian of the linear transformation T that maps the vertices of the cube into those of the parallelepiped. The transformation is required to have the property $T l^i = x^i$ where x^i is the i th point written as column vector and l^i is a column vector having 1 in the i th position and zeroes in the other $n-1$ positions.

Thus $T^{-1} = A$, which is the $n \times n$ matrix whose columns are the vertices of the parallelepipeds. Accordingly the transformation that maps the vertices of the cube into those of the parallelepiped is $Ax' = x$, and its Jacobian is $J = \det A$.

This argument can be extended by use of generalized inverses to the general case of $m (< n)$ points in n dimensional space, so that for linear independent vertices the volume of the parallelepiped is the square root of the determinant of the $m \times m$ matrix $A^T A$. This will now be done for the still more general case in which there may be fewer than m linearly independent vectors.

Let the points be represented as columns of the $n \times m$ matrix A and consider the singular value decomposition, which is available for any matrix. A may be written as

$$A = U Q V^T$$

where U is an $n \times n$ matrix of orthonormal vectors, V is an $m \times m$ matrix of orthonormal vectors and Q is an $n \times m$ matrix whose off diagonal entries are zero. The columns of U are eigenvectors of AA^T ; while the columns of V are eigenvectors of $A^T A$. The diagonal entries of Q are the nonnegative square roots of the eigenvalues. The three matrices are arranged so that eigenvectors and eigenvalues correspond; the non-zero eigenvalues are the same for AA^T and $A^T A$.

A volume may be associated with AA^T or $A^T A$ and the square root used as a volume measure for A .

$$\det AA^T = \det(UQQ^T U^T) = \det QQ^T$$

$$\det A^T A = \det(VQ^T QV^T) = \det Q^T Q$$

since the determinant of an orthonormal matrix is 1 since $U^T U = I$ and $(\det U)^2 = 1$.

The determinant of QQ^T or $Q^T Q$ is simply the product of the non zero eigenvalues times any zero diagonal entries. For the volume measure it is convenient to use the smaller of the two matrices.

If for the smaller matrices, there are k non-zero eigenvalues, then we can say that the points are in a k dimensional subspace and within that subspace, the volume measure for the points is the square root of the product of the non zero eigenvalues.

To find the product of the non-zero eigenvalues, it is of course not necessary to calculate all the eigenvalues. Instead it is sufficient to calculate the left right decomposition of, say $A^T A$: $A^T A = LR$, where L is lower triangular with 1's on the diagonal and R is upper triangular.

Allowing for interchanges, the calculation can restrict any zero diagonal entries in R to the last rows. If there are k non-zero diagonal entries then, we can say that the points are in a k dimensional subspace and that within that subspace the volume measure for the points is the square root of the product of the non-zero diagonal entries in R .

Because of round off, judgment would be needed to recognize when a diagonal entry in R is truly zero. Were this to occur, the calculation of the eigenvalues would be specially useful since the eigenvalues give the individual dimensions of the solid that contains the points. It is equally useful to learn that such a solid is extremely thin in one dimension as to learn that it is of zero thickness.

1.2.2 Convex Hulls

This is the underlying concept for both the linear discrimination between sets of test points having different stability conditions, as well as the recognition of irrelevant test points.

Intuitively, the convex hull of a finite set of points is the smallest solid with polygonal faces and no reentrant corners that contains all the points. Evidently the vertices of the solid are all members of the set of points.

This concept is of great importance for identification of regions with the same stability condition since it gives mathematical meaning to the belief that the points with common stability condition define a region within which the same stability condition prevails.

Also if test points from two different stability regions are close together, and are difficult to distinguish, the mathematical interpretation can be that the convex hulls of the two regions intersect.

Thus the mathematical concept of convex hulls appears to capture precisely the intuitive meaning of a stability region. However the convex hull is a minimal stability region, whereas the intuitive stability region might be thought to extend to some indefinite distance beyond the test points known to be in the region.

The mathematical definition of a convex hull of a set of points S is the smallest convex set that contains S . A convex set is a set which contains the point $x = a_1 x^1 + a_2 x^2$ where $a_1 + a_2 = 1$, provided the points x^1 and x^2 are in the set. This says simply that the line segment between every pair of points in the set is also in the set. Thus a donut with a hole is not a convex set, while a donut without a hole is.

Here superscripts denote different points, while subscripts as in

$$x = (x_1, x_2, \dots, x_n)$$

denote the coordinates, the space being n dimensional.

A hyperplane is the locus of points x such that an equation like

$$c = a_1 x_1 + a_2 x_2 + \dots + a_n x_n$$

holds. This may be written simply as $c = a^T x$ using vector notation. For other points the right-hand side exceeds c , indicating that those points are in the half space on one side of the hyperplane. For still other points, the right-hand side falls short of c , indicating that those points are in the half space on the other side of the half space.

A hyperplane, defined by a vector a and a scalar c , is said to be a supporting hyperplane of a set if

$$c \leq a^T x$$

holds for all points x of the set, with equality holding in some cases. Thus the set is entirely on one side of the hyperplane, with some points of the set belonging to the hyperplane.

Boundary points of the set may be defined as those points of the set which are in some supporting hyperplane.

A vertex is a member x^0 of the set which cannot be represented as a convex linear combination, such as

$$x^0 = a_1 x^1 + a_2 x^2; \quad 1 = a_1 + a_2; \quad 0 < a_1, a_2 < 1$$

for any other points x^1, x^2 of the set.

An interior point x^0 of the set is a point which may be represented in many ways as a convex linear combination

$$x^0 = a_1 x^1 + a_2 x^2; \quad 1 = a_1 + a_2; \quad 0 < a_1, a_2 < 1$$

of points of the set, with x^1 being any vertex of the set. Thus a line through the point from any vertex has points of the set on both sides of the point in question.

A vertex cannot be an interior point since its own representation on a line through it must leave one coefficient $a_1 = 1$ and the other $a_2 = 0$.

A vertex must be one of the original defining points of the convex hull since it cannot be generated from other points using convex linear combinations.

A supporting hyperplane must contain some of the original defining points since the points of a convex hull which are in a supporting hyperplane can be generated only as intermediate points of lines within the plane, not of lines that pierce the plane, since there are no points on one side of the plane.

An easy criterion to identify at least some of the vertex points in a set is that any point with a coordinate greater, or less than that coordinate of any other point is a vertex. Otherwise the point would be an interior point x with a convex linear representation

$$x = a_1 x^1 + a_2 x^2 + \dots + a_m x^m$$

in terms of other points of the set. But then

$$x_k = \sum_i a_i x_k^i \geq \max_i x_k^i \quad \sum_i a_i = 1 \quad \max_i x_k^i$$

which cannot be since by hypothesis x_k exceeds x_k^i for any i . Similarly for least coordinates.

In case several points have the same maximum coordinate, say $x_k = d$ then they belong to a supporting hyperplane $x_k = d$ and among these, those having some maximum or minimum coordinate are all vertices.

Indeed we may see any vertex with extreme coordinate as a point in a supporting hyperplane.

It is easy to see that linear transformations, of stretching, translation or rotation map straight line segments into straight line segments, so that topological properties such as a point being a vertex, an interior point, or in a supporting hyperplane are all preserved. Such transformations can, however alter which point has an extreme coordinate and could in fact produce all the vertices.

Note that there can be at least 2^n vertices with a maximum or minimum coordinate. Thus in higher dimensional space, a smaller proportion of points in a set are likely to be interior points.

Simplicial coordinates for an n dimensional space are defined in terms of $n+1$ points x^1, x^2, \dots, x^{n+1} . An arbitrary point x has simplicial coordinates a_1, a_2, \dots, a_{n+1} means that

$$x = a_1 x^1 + a_2 x^2 + \dots + a_{n+1} x^{n+1}$$

$$\text{and } 1 = a_1 + a_2 + \dots + a_{n+1}$$

To determine the simplicial coordinates, it is necessary to solve the $(n+1) \times (n+1)$ linear system

$$\begin{aligned} X^T a &= x \\ 1^T a &= 1 \end{aligned}$$

where x^i in rectangular coordinates occupies the i th row of X . The set is solvable if the $(n+1) \times (n+1)$ matrix $\begin{pmatrix} X^T \\ 1^T \end{pmatrix}$ is not singular, that is if the $n+1$ vectors x^i are not coplanar, that is no vector b and scalar d can be found so that $x^T b = d$ can be satisfied by all $n+1$ vectors.

In two dimensions the three points are the vertices of a triangle. The sides of the triangle, extended divide the space outside the triangle into 6 regions. Inside the triangle a_1, a_2, a_3 are positive. Upon crossing each side, one of the coordinates turns negative. Upon leaving the triangle through a vertex, two coordinates turn negative. Since the a_i add to 1, there can be no region wherein all simplicial coordinates are

negative. The indication that a point is a boundary point is that some simplicial coordinate is 0. It is of course a vertex if all simplicial coordinates are 0, except one.

Any point of the convex hull of a set of points x^1, x^2, \dots, x^m can be expressed as a convex linear combination

$$x = a_1 x^1 + a_2 x^2 + \dots + a_m x^m$$

$$\text{where } 1 = a_1 + a_2 + \dots + a_m$$

$$\text{and } 0 \leq a_i \leq 1 \text{ for } i = 1, 2, \dots, m$$

This is true of points generated as linear combinations of some two of the original points. Subsequently if this is true of two points x', x'' then, it is true of a convex linear combination of them

$$x = a' x' + a'' x''$$

$$= a' (a'_1 x_1 + a'_2 x_2 + \dots) + a'' (a''_1 x_1 + a''_2 x_2 + \dots)$$

$$= (a' a'_1 + a'' a''_1) x_1 + (a' a'_2 + a'' a''_2) x_2 + \dots$$

$$\text{since } \sum (a' a'_i + a'' a''_i) = a' + a'' = 1$$

$$\text{and } a' a'_i + a'' a''_i \geq 0$$

$$\text{and therefore } a' a'_i + a'' a''_i \leq 0$$

A hyperplane $c = x^T a$ in n -dimensional space can be defined to contain linearly independent points say x^1, x^2, \dots, x^n . If the n coordinates of these points are written as the rows of an $n \times n$ matrix X then

$$Xa = 1$$

where $\mathbf{1}$ denotes a column vector of all 1's. Once \mathbf{a} is found it is frequently desirable to replace \mathbf{a} by the unit vector $\mathbf{a} / \|\mathbf{a}\|$ with $1 / \|\mathbf{a}\|$ written as the right hand side.

If the points are not linearly independent then the matrix X is non-singular. A plane that is to one side of m points where m may be greater or less than the dimension n of the space, can be expressed as

$$X\mathbf{a} \leq c$$

Similarly a plane that discriminates between a set of points whose coordinates are the rows of X and another set of points whose coordinates are the rows of Y , can be expressed as

$$X\mathbf{a} \leq c \leq Y\mathbf{a}$$

These ideas will be developed further in the discussion of discriminant analysis. They will there be used to discriminate between the test data points exhibiting one stability condition, and those exhibiting a different stability condition.

A different use of these concepts is the identification of redundant points, that is test data points which are in the interior of the convex hull defined by other test data points. Such points are of no use in defining where one stability region ends and another begins. They may therefore be discarded thus reducing the size of the relevant data base.

An interior point \mathbf{x} may be expressed as

$$\begin{aligned} \mathbf{x} &= X^T \mathbf{a} \\ \mathbf{1} &= \mathbf{1}^T \mathbf{a} \\ 0 &\leq a_i \leq 1 \end{aligned}$$

where the X matrix consists of the other data points, written as rows. The data points already known to be redundant may be omitted from X .

This is a linear programming problem, without objective function, that is it is the feasibility problem. It can be solved in the usual way by introducing an artificial objective function representing the violations of the constraints. If this objective function can be reduced to zero then the problem is feasible. Such a device is quite reasonable and economical if upon establishment of feasibility, an authentic objective function is then pursued by the mechanism already in place.

However the artificial objective function is less attractive when feasibility is the only question to be answered. Thus it seemed useful to modify the usual simplex algorithm so as to bypass use of an objective function.

A further compelling reason for doing this is to accommodate the coding to the special situation here, where a test point x once found to be non-redundant would need to be considered for the basis used in evaluating subsequent test points for redundancy.

To take advantage of these two aspects of this particular application, a special modification of the simplex algorithm has been developed. It has been named the "Non-Objective Simplex Method."

1.2.3 The Non-Objective Simplex Method

The problem addressed is feasibility of the linear programming problem:
find x such that

$$Ax = b \quad x \geq 0$$

where A is an $m \times n$ matrix where $m < n$, x is a vector of n entries and b has m entries. Feasibility asks whether there are any x vectors meeting these requirements.

The objective function which might accompany this problem is not relevant to the feasibility question. However it is usual to introduce an artificial objective function and artificial variables so that the augmented problem is trivially feasible and if its (artificial) objective function can be driven sufficiently large, feasibility of the original problem will have been established.

There are two disadvantages of the usual approach. It enlarges the matrix A and feasibility is only obliquely addressed and therefore perhaps inefficiently. Its advantage is that the code designed to solve the linear programming problem with objective function is also used for the prior task by finding a feasible starting point. But this is no advantage if only feasibility need be shown.

Thus it is believed that the approach to be described may prove more efficient for the case in which the question of feasibility is the only task.

Since this development was done separately from other work being reported on, the notation is somewhat different.

The columns of A will be denoted as a^1, a^2, \dots, a^n . A will be assumed to be of full rank m . The vector x is a set of coefficients for the columns of A , expressing b as a linear combination of the columns of A . Since any vector can be expressed in terms of only m columns of A ,

it will be assumed that only m components of x are non-zero. Feasibility is established if an x is found having no negative coefficients. The feasible solution will be found by generating a sequence of non-feasible solutions x^1, x^2, \dots which hopefully will have fewer and fewer negative coefficients.

Initially x^1 is gotten by picking m linearly independent columns of A , and solving for the linear combination x^1 by which they represent b . Denote by n_1 the number of negative coefficients in x^1 . The next goal is to find an x^2 which will have fewer negative coefficients.

To do this, it is convenient to represent all the columns of A as linear combinations of the basis columns. These representations are collected in a matrix E , whose ij entry is the coefficient of the i th basis vector in the representation of the j th column of A . If the j th column is a member of the basis, say the k th member, then column j of E is all zeros except that e_{kj} is 1. This matrix E is essentially the "tableau" spoken of in linear programming literature.

Thus if B consists of the m columns of A which constitute the basis then

$$E = B^{-1}A \quad x^1 = B^{-1}b$$

so that the columns of A in the basis become columns of the identity in E .

Choice of Advantageous Change to Basis

Consider what advantage there might be in replacing some member of the basis used in x^1 by column k of A .

If the basis consists of columns numbered c_1, c_2, \dots, c_m of A then

$$a^k = \sum e_{ik} a^{c_i}$$

and is represented in E by $[e_{1k}, e_{2k}, \dots]^T$ as the k th column. The representation of b using x^1 is

$$b = \sum_{i=1}^m x_i^1 a^{ci}$$

but is also trivially equal, for any multiplier r , to

$$\begin{aligned} b &= \sum_{i=1}^m x_i^1 a^{ci} - r \sum_{i=1}^m e_{ik} a^{ci} + r a^k \\ &= \sum_{i=1}^m (x_i^1 - r e_{ik}) a^{ci} + r a^k \end{aligned}$$

To represent b in terms of a basis containing a^k in place of a^{cj} , all that is needed is to set

$$r = x_j^1 / e_{jk}$$

assuming that $e_{jk} \neq 0$. The fact that e_{jk} is not 0 assures that the new basis would still have no more than m linearly independent members.

Since the coefficients in x^2 are $x_i^1 - r e_{ik}$ and r it is easy to count how many are negative and therefore whether introduction of column k into the basis can be advantageous.

If $e_{ik} > 0$ and $r < x_i^1 / e_{ik}$ then $x_i^2 > 0$

If $e_{ik} < 0$ and $x_i^1 / e_{ik} \leq r$ then $x_i^2 \geq 0$

If $e_{ik} = 0$ then $x_i^2 = x_i^1$

Since if $r \geq 0$ then $x_k^2 \geq 0$ the lower bounds on r should also include 0.

Denote the least and greatest of the lower bounds on r , (including 0) by r_1 and r_2 , respectively; denote the least and greatest of the upper bounds by r_3 and r_4 , respectively. By their definition

$$r_1 \leq r_2 \text{ and } r_3 \leq r_4$$

If $r_2 \leq r_3$ then all bounds on r will be satisfied if k displaces either the basis vector determining r_2 or that determining r_3 . The number of negative coefficients in x^{22} would then be

$$n^2(k, r) = \text{number of } x_i^1 < 0 \text{ for which } e_{ik} = 0$$

If $r_1 < r_3 < r_2 < r_4$, then some bounds on r are violated and r is best chosen in the interval

$$r_3 < r < r_2$$

since choice of r outside this interval would violate upper or lower bounds without a compensating satisfaction of lower or upper bounds.

If $r_3 < r_1 < r_2 < r_4$ or $r_1 < r_3 < r_4 < r_2$, r should again be chosen in the interval

$$r_3 \leq r \leq r_2$$

as there is only disadvantage outside this interval.

If $r_3 < r_1 < r_4 < r_2$ or $r_3 < r_4 < r_1 < r_2$, r is again to be chosen in the interval

$$r_3 \leq r \leq r_2$$

which here says to examine all possibilities.

Thus in all cases r is to be chosen in the closed interval extending between r_2 and r_3 . It is in general necessary to search for all bounds in the interval to see how many are violated. The only values of r to be evaluated are the values equal to a bound. Then

$$n^2(k, r) = [\text{number of upper or lower bounds which are violated}] \\ + [\text{number of } x_i^1 < 0 \text{ for which } e_{ik} = 0]$$

This procedure is done for all k , or at least until an n^2 is found which is less than n^1 .

If at the best value of r , there is only one bound, say for the basis vector number c_p , which is column, say h , of A , then column a^k of A replaces column a^h in the basis and the c_p th basis vector becomes column a^k , rather than column a^h .

In case several bounds are simultaneously satisfied at the best value of r , k replaces only one of them in the basis - which one may be chosen arbitrarily. Thus the basis still has m linearly independent vectors even though b can be expressed with fewer.

Updating x and E

It is clear from the foregoing how x^2 is computed from x^1

$$x_i^2 = x_i^1 - r e_{ik} \text{ for } i \neq p$$

$$x_p^2 = 0$$

Also E used for the basis associated with x^1 , which can be labeled E^1 , can be easily updated to the matrix E^2 associated with the basis associated with x^2 .

Column a^1 of A was represented by the j th column of E^1 :

$$a^j = \sum_{i=1}^m e_{ij} a^{ci}$$

In particular

$$a^k = \sum_{i=1}^m e_{ik} a^{ci}$$

which implies

$$a^{cp} = - \sum_{\substack{i=1 \\ i \neq p}}^m e_{ik} / e_{pk} a^{ci} + 1/e_{pk} a^k$$

where e_k was not 0 since the basis vector c_p replaced by column k was a bound for r and so had its e_{pk} non zero.

Replacing a^{c_p} in the equation for a^j , gives

$$a^j = - \sum_{\substack{i=1 \\ i \neq p}}^m [e_{ij} - e_{pj} e_{ik} / e_{pk}] a^{c_i} + (e_{pj} / e_{pk}) a^k$$

Thus

$$e_{ij}^2 = e_{ij}^1 - e_{pj}^1 e_{ik}^1 / e_{pk}^2 \quad \text{for } i \neq p$$

$$e_{pj}^2 = e_{pj}^1 / e_{pk}^1$$

Note that this implies that column k of E^2 is all 0, except for e_{pk}^2 which is 1.

Also whereas in E^1 column h had been all 0 except for e_{pk}^1 which was 1, in E^2 this column becomes, according to the above formula

$$e_{ih}^2 = - e_{ik}^1 / e_{pk}^1 \quad \text{for } i \neq p$$

and

$$e_{ph}^2 = 1 / e_{pk}^1$$

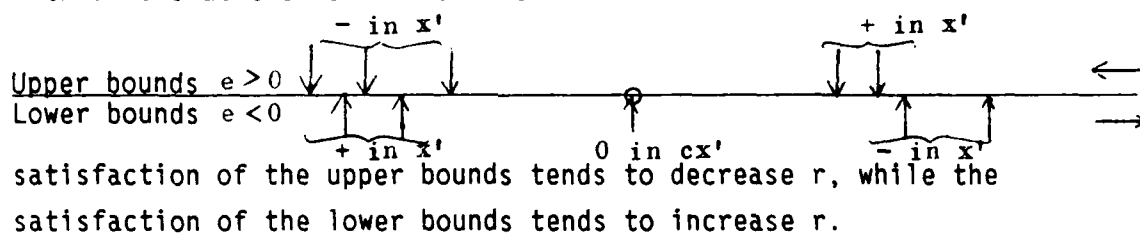
Other columns, say column q , which are in both the basis associated with x^1 , as well as the basis associated with x^2 retain their form - all 0 except for a single 1 - since e_{iq}^1 was 0 in E^1 being the coefficient of a^{c_p} in the representation of a^q which was also in the basis. Thus $e_{iq}^2 = e_{iq}^1$.

We see therefore that all E matrices contain an $m \times m$ identity matrix, in permuted form. By recording which columns of A correspond to which columns of the remaining columns of E , as well as which correspond to the rows of E , it is possible to omit storage of the $m \times m$ identity.

The procedure will have found a feasible point once all coefficients of x have been made non negative, i.e. once n_j has become 0.

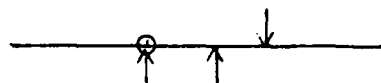
Further Discussion of Change to Bases

It is easy to observe schematically what difficulties can arise in the choice of r , by use of the following schematic. A line representing values of r ranging from negative to positive values is marked on the upper side with arrows at the upper bounds on r , and on the lower side with arrows at the lower bounds on r . The

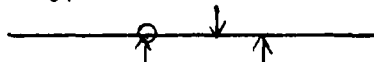


The last point x^1 is represented by $r = 0$, where the lower bound on the new column k is located.

If r is increased from 0 and the first bound it encounters is a lower bound then that choice of r eliminates a negative coefficient and includes a new positive coefficient, leaving other coefficients of x unchanged in sign.



If however the first bound was in upper bound, that choice of r represents no net change in the types of coefficients of x , i.e. +, 0, -.



If r was decreased and the first bound encountered, other than that at 0, was an upper bound, that choice of r represents no net change in the type of coefficient.



If r was decreased and the first bound encountered, other than at 0, was a lower bound, that choice of r represents a switch of a + coefficient to a - coefficient.



Thus the simple strategy of choosing r at an upper or lower bound nearest $r = 0$, which represents x^1 , increases the + coefficients in x^2 over those in x^1 only if r is increased and the first bound encountered is a lower bound.

It is of course a less complex procedure simply to search the full range of bounds on r .

As to which k column is most advantageous for inclusion, one criterion is to choose at least initially the column in E which has most uniformity in sign, since then most bounds in 0 are either upper or lower bounds, and can easily be satisfied en masse.

Program Flow

- o Choose basis among vectors with extreme values max, min of any component.
- o Form triangular decomposition, modifying choice of vectors as decomposition proceeds.
- o Record basis names in vector IR.
- o Apply triangular factors to columns of A not in basis. Form matrix E . Keep running count of sign diversity. Put vectors with extreme value first. Record column number of non basic vectors in vector IC.
- o If all entries in column have same sign + record for inclusion. If all entries have sign - then a feasible point has been found.

o Apply factors to b to get x^1 . Record number of minus signs in n .

o Update E , x , IR , IC . Repeat preceding step.

o Calculate for each k

lower and upper bounds, $0_1, 0_2, 0_3, 0_4$
pick the 0 with the maximum advantage

Quit if all $x = 0$.

o Calculate for each k : lower and upper bounds r_1, r_2, r_3, r_4 .
Pick the r with the maximum advantage.

o Update E , x , IR , IC . Repeat preceding step.

o Quit if all x non-negative.

1.2.4 Discriminant Analysis

The task addressed by discriminant analysis is to find a surface that separates two sets of points. If the coordinates of a point are substituted into the mathematical equation of the surface: $f(x_1, x_2, \dots) = C$ the point is on one side of the surface if the result exceeds C, is on the other side if the result falls short of C and is exactly on the surface if the result equals C. The surface is said to discriminate between the two sets of points.

The simplest type of surface is the plane, or as is sometimes said for higher dimensional space, the hyperplane. The equation for a plane is of the form: $\sum a_i x_i = C$, that is, it is a linear function of the x_i as well as a linear function of the coefficients a_i .

It is possible to devise methods for nonlinear expressions. For example, the ellipse in two dimensions has the form: $a_1 x_1^2 + a_2 x_2^2 = C$. However, the methods for linear expressions can address this problem by considering the expression as linear in the transformed variables x_1^2 and x_2^2 .

This device cannot treat expressions in which the unknown coefficients do not appear linearly; such as $a_1 x_1^{a_2} + a_3 x_2^{a_4} = C$ or $a_1 \sin(a_2 x_1) + a_3 \sin(a_4 x_2) = C$.

Discriminant analysis can be viewed as geometric or as probabalistic. The geometric view is expressed above. The probabalistic view seeks a surface which discriminates with a high confidence. It can address situations in which the sets of points substantially interpenetrate one another.

For the flutter prediction problem, it is believed that linear, geometric discriminant analysis is the basic tool and that it will prove adequate, especially when enhanced with nonlinear combination variables. Any interpenetration of stability regions is believed attributable to imprecision of measurement or to higher order effects not modeled here. In either case, the degree of interpenetration is expected to be small.

The empirical flutter prediction problem is directly addressed by discriminant analysis. The Annular Cascade Data Base consists of a large number of test points each identified as being in one out of fourteen possible stability regions. Taking these regions two at a time, it is desired to find a plane that separates each region from the other. To conclude that a test point is in a stability region, it would, ideally, need to be on the side of each of the thirteen planes that discriminated that stability region.

GALACTIC contains two linear, geometric discriminant analysis methods. Both utilize linear programming.

The first method addresses the easier question: whether there is a discriminating plane. This question can be answered without actually exhibiting the plane. The second method addresses the more difficult problem of finding the plane. The first method will be called the Discriminant Feasibility method.

The first method is valuable because it can identify the smallest number of variables for which the sets are separable. The first method is quick and robust. The second method which is more costly and less robust can therefore avoid unprofitable exploration of small subspaces.

The second method has been modified to become a third method, called the Relaxed Discriminant method. It seeks to find a pair of parallel planes that discriminate the stability regions. Between the planes there is either a gap or an overlap. Thus the third method should provide an approximate solution even when the regions are not truly separable.

Consider that there are two sets of points in m dimensional space, one with n_1 points and the other with n_2 points. These points can be represented respectively as the rows of an $n_1 \times m$ matrix X_1 and the rows of an $n_2 \times m$ matrix X_2 .

Discriminant Feasibility Method

The first method will simply determine whether the two sets of points intersect, in the sense that their convex hulls intersect. In two dimensions the convex hull is the smallest polygonal region that contains all the points and has no reentrant corners. Algebraically a point z is in the convex hull of the n_1 points in X_1 if z can be represented as a linear combination of the rows of X_1

$$z = aX_1 \quad a_i \geq 0$$

where a is a row vector whose components are non negative and which add to 1

$$\sum_{i=1}^n a_i = 1$$

Now z is in both sets if there is also a row vector b such that

$$z = b X_2$$
$$\sum_{i=1}^{n_2} b_i = 1 \quad b_i \geq 0$$

The problem of determining whether there are points z which are in both convex hulls may be converted to a standard linear programming problem as follows. Equate the two expressions for z :

$$aX_1 - bX_2 = 0$$

or

$$(a \ b) \begin{pmatrix} X_1 \\ -X_2 \end{pmatrix} = 0$$

The $\sum a_i = 1$ and $\sum b_i = 1$ requirements can be incorporated

$$(a \ b) \begin{pmatrix} X_1 & 1 & 0 \\ & 1 & \\ & 1 & \\ \text{---} & 0 & 1 \\ -X_2 & 1 & 1 \end{pmatrix} = (0 \dots 1 \ 1)$$

Combining a and b into $y = (a, b)$, this equation becomes

$$\begin{pmatrix} X_1 & -X_2 \\ 11\dots & 00\dots \\ 00\dots & 11\dots \end{pmatrix} y = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 1 \end{pmatrix} ; y \geq 0$$

Thus it is required to find an n entry non-negative column vector y which satisfies the above $m+2$ equality constraints.

The objective function is of secondary importance since the basic question here is whether there is a feasible solution, that is, whether the two convex hulls intersect. If they do, then the two sets cannot be discriminated linearly.

Since linear programming codes generally require an objective function $C_i y_i$, it is convenient to pick the C_i so as to emphasize the role of certain points in X_1 or X_2 .

However, the easy choice of making the C_i for X_1 equal, and those for X_2 equal is not advisable since in view of the constraints $\sum a_i = 1, \sum b_i = 1$, such an objective function would reduce to $C_1 + C_{n+1}$ which is constant. While inconsequential for the feasibility question, a constant might inadvertently cause programming problems.

Thus to conclude that two sets cannot be linearly discriminated, it is sufficient to show that a certain linear programming problem is feasible. Conversely, if the linear programming problem is not feasible, then the convex hulls of the two sets do not intersect and it is therefore possible

to find a plane that lies between them, intersecting neither set. The plane is used to discriminate between the two sets. Note that this method does not produce such a discriminating plane.

Determination of the Discriminating Plane

The second method is more complicated, but it accomplishes more. It identifies a discriminating plane, that is, a plane that separates two non-intersecting sets.

Any point x that satisfies $x p = d$ is on a plane having column vector p and perpendicular from the origin and situated a distance $d / \|p\|^2$ from the origin.

Such a plane would discriminate the sets in X_1 and X_2 if

$$X_1 p \geq d$$

$$X_2 p \leq d$$

or equivalent,

$$\begin{pmatrix} X_1 & -1 \\ & -1 \\ & \vdots \\ -X_2 & +1 \\ & +1 \\ & \vdots \end{pmatrix}$$

$$\begin{pmatrix} p \\ d \end{pmatrix} \geq 0$$

or abbreviating

$$Z q \geq 0$$

where Z is $n^*(m+1)$, n being n_1+n_2 , and q is an $m+1$ vector.

A few comments are appropriate.

There is no inequality constraint on the components of p or on d ; hence it is no restriction to choose that $X_1 p$ exceed d and for $X_2 p$ to be less than d , rather than vice versa.

The problem as stated allows the two sets, and therefore the discriminating plane, to have common boundary points. This could be prohibited by requiring

$$X_1 p \geq d_1 ; \quad X_2 p \leq d_2 ; \quad d_1 - d_2 \geq k$$

or $Z q \geq t$

for some chosen $k \geq 0$. Z would have one extra column as well as an extra row for the $d_1 - d_2 - k$ constant. If alternatively k were allowed to be slightly negative, then some overlap between the two sets would be allowed.

While the problem statement involves linear inequalities, it is not in standard linear programming format since there are no inequalities constraining q . The following converts the problem to standard format.

$$\text{Write} \quad Zq = t \quad t \geq 0$$

and utilize the so called generalized inverse Z^+ for Z

$$q = Z^+ t.$$

Digression on Generalized Inverses

A digression about generalized inverses is appropriate here. First it will be explained how best to compute the generalized inverse, and then what is its usefulness.

The best way to compute Z^+ is to compute the singular value decomposition of Z . This is a representation of the $n \times m$ matrix Z as the product

$$Z = U Q V^T$$

where U is $n \times n$ and orthogonal ($U^T = U^{-1}$), V is $m \times m$ and orthogonal ($V^T = V^{-1}$) and Q is an $n \times m$ matrix which is zero except for the diagonal terms which may or may not be 0. The generalized inverse Q^+ of Q is defined as the $m \times n$ matrix which is zero except possibly for the diagonal entries. The ii entry is 0 if $q_{ii} = 0$ and is $1/q_{ii}$ if $q_{ii} \neq 0$.

Thus $I_0 = Q Q^+$ is an $n \times n$ matrix which is zero except that its ii diagonal entry is 1 when $q_{ii} \neq 0$.

Now the generalized inverse of Z is

$$Z^+ = V Q^+ U^T$$

As a result,

$$ZZ^+ = U Q V^T V Q^+ U^T$$

$$U Q Q^+ U^T$$

$$U I_0 U^T$$

If for instance it were square and non-singular, then Q would have no zeroes on its diagonal and accordingly I_0 would also not have zeroes on its diagonal so that $ZZ^+ = U I U^T = I$ and Z^+ is the (usual) inverse Z^{-1} .

If $Zq = t$ represented a least squares problem in which $n \geq m$ and Z had rank m , then the usual solution is

$$q = (Z^+ Z)^{-1} Z^+ t$$

which is

$$\begin{aligned} &= (V Q^T U^T U Q V^T)^{-1} V Q^T U^T t \\ &= (V Q^T Q V^T)^{-1} V Q^T U^T t \\ &= (V^T)^{-1} (Q^T Q)^{-1} V^{-1} V Q^T U^T t \\ &= V (Q^T Q)^{-1} Q^T U^T t \end{aligned}$$

But since Z had rank m , then Q could not have zeroes on its diagonal, $Q^T Q$ is an $m \times m$ non-singular diagonal matrix and $(Q^T Q)^{-1} Q^T = Q^+$. Thus the usual least squares solution is also the generalized inverse

$$q = VQ^+U^T t$$

Other situations which have been treated in traditional ways could also be presented as persuasive of the reasonability and practicality of the stated generalized inverse. However, it is perhaps more useful now to give the following derivation.

Since any matrix whether square or rectangular, singular or non-singular, has a singular value decomposition, the problem $Zq = t$ can be written equivalently as

$$UQV^T q = t$$

or

$$QV^T q = U^T t$$

or

$$Qr = s$$

where $s = U^T t$ and q can be gotten from r by

$$q = Vr$$

The equation $Qr = s$, spelled out is simply

$$q_{11}r_1 = s_1$$

$$q_{22}r_2 = s_2$$

⋮

If $q_{ij} \neq 0$, then $r_i = s_i/q_{ij}$. But if $q_{ij} = 0$ then r_i is arbitrary. According to the generalized inverse defined above, r_i is set to 0 when $q_{ij} = 0$. This says that

$$r = Q^+s$$

But then

$$\begin{aligned} q &= Vr \\ &= VQ^+s \\ &= VQ^+U^Tt \end{aligned}$$

which is the formula presented above.

Determination of the Discriminating Plane (continued)

For the discrimination question, it is necessary to ask when the generalized inverse is an exact solution. This is the case provided that Zq equals t , that is, provided $ZZ^+t = t$ or

$$\begin{aligned} [Z(VQ^+U^T) - I] t &= 0 \\ [UQV^TVQ^+U^T - I] t &= 0 \\ [UQQ^+U^T - I] t &= 0 \end{aligned}$$

Here QQ^+ is an $n \times n$ diagonal matrix, say D whose ii entry is 1 if $q_{ij} \neq 0$ and 0 if $q_{ij} = 0$. Thus the above may be written

$$[UDU^T - I] t = 0$$

or

$$[U(D-I)U^T] t = 0$$

or

$$UI_1 U^T t = 0$$

Since U is non-singular, this may be written simply as

$$I_1 U^T t = 0$$

where I_1 is an $n \times m$ diagonal matrix whose ii entry is 0 if $q_{1i} \neq 0$ and +1 if $q_{1i} = 0$. This is the necessary and sufficient condition that

$$q = VQ^+ U^T t$$

is an exact solution for $Zq = t$. This might seem to be a virtually impossible condition to meet. Indeed when Z is rectangular and of rank less than m , as in a least square problem, it is not likely that q can be found which will exactly satisfy $Zq = t$ for a preassigned t .

But here the t is not preassigned; any t with non-negative components will do. Indeed t is simply an eigenvector of $UI_1 U^T$ which corresponds to a 0 eigenvalue. The necessary and sufficient condition $UI_1 U^T t = 0$ can be simplified to

$$I_1 U^T t = 0$$

since U is square and non-singular.

For the discriminant analysis task at hand, finding a q such that Zq is non-negative is seemingly effortlessly satisfied by choosing a non-negative t (that satisfies any other constraints) and calculating

$$q = Z^+ t$$

provided that this implies $Zq = t$. This implication is valid if

$$ZZ^+ t = t$$

which is developed above.

The problem of finding a discriminating plane between two sets has thus been reduced to the following linear programming problem in standard format: find t such that

$$I_1 U^T t = 0 \quad t \geq 0$$

This is a system of at least $n-m$ equations in n unknowns: t_1, t_2, \dots, t_n .

The solution of the system of equations can actually be expressed as

$$t = UI_2 c$$

where c is arbitrary and $I_2 = I - I_1$ which is D defined above. To show this, substitution gives

$$I_1 U^T UI_2 c = I_1 (I - I_1) c = 0$$

Thus t is a linear combination of the rows of U^T which are not in the system $I_1 U^T t = 0$.

Clearly then the constraint $I_1 U^T t = 0$ describes an unbounded region. If any vector in this region satisfies $t \geq 0$, an arbitrarily large multiple of the vector also is feasible. Thus the feasibility region if it exists is unbounded.

The linear programming problem thus posed has a finite solution only if the objective function seeks small rather than large feasible points. But seeking small feasible points leads to the origin which is clearly a feasible point. Thus the linear programming problem leads to either ill defined infinite solutions or the trivial solution.

To avoid the dilemma, a constraint

$$\sum t_i = 1$$

may be included, together with an objective function such as

$$\text{maximize } \sum t_i$$

Since the feasible region if it exists had been unbounded, this objective function will locate the optimum on the new constraint, always giving 1 as the value of the objective function.

The objective function is, however, not without meaning. For a point in the first set, its t is $x_p - d$, while for a point in the second set, its t is $d - x_p$, according to $x_2 p \leq d \leq x_1 p$. This says that t is the perpendicular distance of the point from the plane. Thus the stated objective function is the sum of the perpendicular distance of the n points from the plane. The points in one set being on one side of the plane; and the points in the second set being on the other side. Thus the objective function seeks to maximize, in this sense, the distance of the sets from the plane.

If the linear programming problem does not have a solution, then there is no plane that separates the two sets.

If the linear programming problem does have a solution t , then

$$q = VQ^+U^T t$$

The first m components of q constitute the vector p , and the last component, the $(m+1)$ st is the distance d .

This p and d satisfy

$$x_1 p \geq d, \quad x_2 p \leq d$$

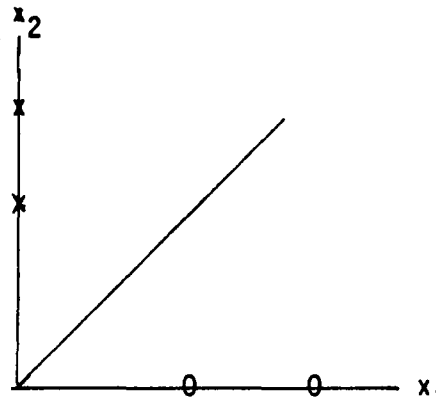
so that the plane $x p = d$ discriminates between the two sets.

Example 1: Discriminating planes exist

$$\begin{aligned} n_1 &= 2 & m &= 2 \\ n_2 &= 2 \\ n &= 4 \end{aligned}$$

$$X_1 = \begin{bmatrix} 2 & 0 \\ 3 & 0 \end{bmatrix}$$

$$X_2 = \begin{bmatrix} 0 & 2 \\ 0 & 3 \end{bmatrix}$$



The points in X_1 are denoted by 0, and those in X_2 by X.

To apply the first method, set up the LP problem: find $y \geq 0$ such that

$$\begin{bmatrix} 2 & 3 & 0 & 0 \\ 0 & 0 & -2 & -3 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

There is a unique solution to the equality for y

$$y = \begin{bmatrix} 3 \\ -2 \\ 3 \\ -2 \end{bmatrix}$$

which, however, violates the requirement $y \geq 0$. Thus the LP problem is non feasible and according to the first method the points can be linearly discriminated. Note how easy the first method is, but how minimal is the information it supplies.

The second method requires finding p and d so that

$$\begin{pmatrix} 2 & 0 & -1 \\ 3 & 0 & -1 \\ 0 & -2 & 1 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ d \end{pmatrix} \geq 0$$

Some obvious solutions are

$$p = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \quad -\sqrt{2} \leq d \leq \sqrt{2}$$

Solutions can be produced systematically as follows. The singular value decomposition of the above matrix is

$$\begin{pmatrix} 2 & 0 & -1 \\ 3 & 0 & -1 \\ 0 & -2 & 1 \\ 0 & -3 & 1 \end{pmatrix} = UQV^T$$

where

$$U = \begin{pmatrix} .41884514 & -.39223227 & .56970935 & .58834842 \\ .56970932 & -.58834834 & -.41884514 & -.39223228 \\ -.41884514 & -.39223226 & -.56970937 & .58834839 \\ -.56970935 & -.58834839 & .41884514 & -.39223226 \end{pmatrix}$$

$$Q = \begin{pmatrix} 4.1087133 & 0 & 0 \\ 0 & 3.6055510 & 0 \\ 0 & 0 & .34419860 \\ 0 & 0 & 0 \end{pmatrix}$$

$$V = \begin{pmatrix} .61985781 & -.70710676 & -.34025909 \\ .61985783 & .70710675 & -.34025909 \\ -.48119902 & .13512450E-7 & -.87661135 \end{pmatrix}$$

as computed by the IMSL subroutine LSVDF. The solution is

$$\begin{pmatrix} p \\ d \end{pmatrix} = VQ^+U^T t$$

where t solves

$$I_1 U^T t = 0 \quad t \geq 0$$

$$\text{where } I_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Thus t solves

$$(-.58834842 + .39223228 \quad -.58834839 + .39223226) \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = 0$$

which allows arbitrary choice for t_1, t_2, t_3 whereupon t_4 is determined by

$$t_4 = (3/2) t_1 - t_2 + (3/2) t_3$$

However, the condition that $t \geq 0$ requires that

$$(3/2) (t_1 + t_3) \geq t_2 \geq 0$$

To get $\begin{pmatrix} p \\ d \end{pmatrix} = VQ^+U^T t$, the equation for Q^+ is

$$Q^+ = \begin{pmatrix} .2433852 & 0 & 0 & 0 \\ 0 & .2773501 & 0 & 0 \\ 0 & 0 & 2.9052994 & 0 \end{pmatrix}$$

so that

$$VQ^+ = \begin{pmatrix} .15086421 & -.19611613 & -.98855428 & 0 \\ .15086421 & .19611613 & -.98855428 & 0 \\ -.11711671 & .37476796E-8 & -2.5468183 & 0 \end{pmatrix}$$

$$VQ^+U^T = \begin{pmatrix} -.42307674 & .61538439 & .5769228 & -.3846161 \\ -.5769228 & .3846152 & .4230767 & -.6153844 \\ -1.4999998 & .9999999 & 1.4999998 & -.9999999 \end{pmatrix}$$

The more exact answer is

$$VQ^+U^T = 1/26 \begin{pmatrix} -11 & 16 & 15 & -10 \\ -15 & 10 & 11 & -16 \\ -39 & 26 & 39 & -26 \end{pmatrix}$$

so that

$$\begin{pmatrix} p_1 \\ p_2 \\ d \end{pmatrix} = VQ^+U^T \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix}$$

$$= 1/26 \begin{pmatrix} -11 & 16 & 15 & -10 \\ -15 & 10 & 11 & -16 \\ -39 & 26 & 39 & -26 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ 3/2 t_1 - t_2 + 3/2 t_3 \end{pmatrix}$$

$$= 1/26 \left(t_1 \begin{pmatrix} -26 \\ -39 \\ -78 \end{pmatrix} + t_2 \begin{pmatrix} 26 \\ 26 \\ 52 \end{pmatrix} + t_3 \begin{pmatrix} 0 \\ -13 \\ 0 \end{pmatrix} \right)$$

$$= t_1 \begin{pmatrix} 1 \\ 1.5 \\ 3 \end{pmatrix} + t_2 \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} + t_3 \begin{pmatrix} 1 \\ -0.5 \\ 0 \end{pmatrix}$$

As a check, compute

$$\begin{pmatrix} 2 & 0 & -1 \\ 3 & 0 & -1 \\ 0 & -2 & 1 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ d \end{pmatrix} = t_1 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1.5 \end{pmatrix} + t_2 \begin{pmatrix} 0 \\ 1 \\ 0 \\ -1 \end{pmatrix} + t_3 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1.5 \end{pmatrix}$$

The components are all obviously non-negative except for the last which is

$$3/2 (t_1 + t_3) - t_2$$

and non-negative by choice of t_2 in the range

$$3/2 (t_1 + t_3) \geq t_2 \geq 0$$

While the singular value decomposition is the more useful general approach to computing generalized inverses, the generalized inverse can sometimes be computed more simply and exactly as

$$Z^+ = (Z^+ Z)^{-1} Z^T$$

Thus in example 1

$$Z^+ = \begin{pmatrix} 13 & 0 & -5 \\ 0 & 13 & -5 \\ -5 & -5 & 4 \end{pmatrix}^{-1} Z^T$$

$$= 1/26 \begin{pmatrix} 27 & 25 & 65 \\ 25 & 27 & 65 \\ 65 & 65 & 169 \end{pmatrix} Z^T$$

$$= 1/26 \begin{pmatrix} -11 & 16 & 15 & -10 \\ -15 & 10 & 11 & -16 \\ -39 & 26 & 39 & -26 \end{pmatrix}$$

Hence the equation for t is

$$0 = (ZZ^+ - I) t$$

$$= \left[\frac{1}{26} \begin{pmatrix} 17 & 6 & -9 & 6 \\ 6 & 22 & 6 & -4 \\ -9 & 6 & 17 & 6 \\ 6 & -4 & 6 & 22 \end{pmatrix} - I \right] t$$

$$= \frac{1}{26} \begin{pmatrix} -9 & 6 & -9 & 6 \\ 6 & -4 & 6 & -4 \\ -9 & 6 & -9 & 6 \\ 6 & -4 & 6 & -4 \end{pmatrix} t$$

which reduces to the single equation

$$0 = -(3/2)t_1 + t_2 - (3/2)t_3 + t_4$$

or

$$t_4 = -(3/2)t_1 - t_2 + (3/2)t_3$$

The condition that $t \geq 0$ is fulfilled if t_1 and t_2 are arbitrarily chosen as non-negative, t_2 satisfies

$$(3/2)(t_1 + t_3) \geq t_2 \geq 0$$

and

$$t_4 = (3/2)(t_1 + t_3) - t_2$$

From this, the equation for p and d is

$$\begin{pmatrix} p_1 \\ p_2 \\ d \end{pmatrix} = Z^+ t$$

$$= 1/26 \left(t_1 \begin{pmatrix} 26 \\ 39 \\ 78 \end{pmatrix} + t_2 \begin{pmatrix} 26 \\ 26 \\ 52 \end{pmatrix} + t_3 \begin{pmatrix} 0 \\ -13 \\ 0 \end{pmatrix} \right)$$

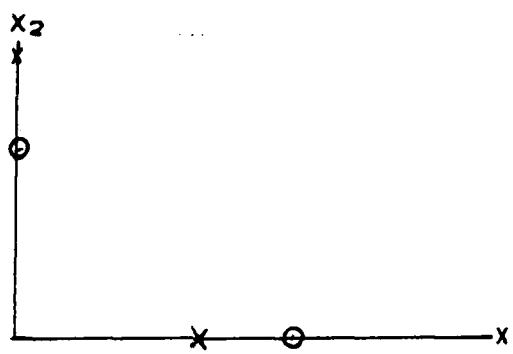
$$= t_1 \begin{pmatrix} 1 \\ 1.5 \\ 3 \end{pmatrix} + t_2 \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} + t_3 \begin{pmatrix} 0 \\ -0.5 \\ 0 \end{pmatrix}$$

Example 2: Discriminating planes exist

$$\begin{aligned} n_1 &= 2 & m &= 2 \\ n_2 &= 2 \\ n &= 4 \end{aligned}$$

$$\begin{aligned} X_1 &= \begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix} \\ X_1 &= \begin{bmatrix} 3 & 0 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} X_2 &= \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \\ X_1 &= \begin{bmatrix} 0 & 3 \end{bmatrix} \end{aligned}$$



To apply the first method, set up the LP problem: find $y \geq 0$ such that

$$\begin{pmatrix} 0 & 3 & -2 & 0 \\ 2 & 0 & 0 & -3 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} y = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

There is a unique solution to the equality for y

$$y = (1/5) \begin{pmatrix} 3 \\ 2 \\ 3 \\ 2 \end{pmatrix}$$

which satisfies the requirement $y \geq 0$. Thus the LP problem is feasible and according to the first method the points cannot be linearly discriminated. Nonetheless, the second method will be attempted to show the manner in which it will fail.

The second method requires finding p and d so that

$$\begin{pmatrix} 0 & 2 & -1 \\ 3 & 0 & -1 \\ -2 & 0 & 1 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ d \end{pmatrix} \geq 0$$

No non-trivial solution is possible since the inequalities are

$$3p_1 \geq d \geq 2p_1$$

$$2p_2 \geq d \geq 3p_2$$

The first requires that $p_1 \geq 0$ and $d \geq 0$ while the second requires that $p_2 \geq 0$ and $d \geq 0$. Thus $p_1 = p_2 = d = 0$.

The singular value decomposition of the above matrix is given as

$$\begin{pmatrix} -2 & 3 & 0 & 0 \\ 0 & 0 & 2 & -3 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} = UQV^T$$

where

$$U^T = \begin{pmatrix} -.41884516 & .56970937 & .41884512 & -.56970932 \\ -.39223222 & .58834834 & -.39223229 & .5883484 \\ .56970932 & .41884512 & -.56970934 & -.41884515 \\ -.58834842 & -.39223228 & -.58834839 & -.39223226 \end{pmatrix}$$

$$Q = \begin{pmatrix} 4.108733 & 0 & 0 \\ 0 & 3.6055508 & 0 \\ 0 & 0 & .34419858 \\ 0 & 0 & 0 \end{pmatrix}$$

$$V = \begin{pmatrix} .61985786 & .70710670 & .34025909 \\ .61985777 & -.70710675 & .34025908 \\ -.48119902 & .21704414E-7 & .87661133 \end{pmatrix}$$

as computed by the IMSL subroutine LSVDF. The solution is

$$\begin{pmatrix} p \\ d \end{pmatrix} = VQ^+U^T t$$

where t solves

$$I_1 U^T t = 0 \quad t \geq 0$$

$$\text{where } I_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Thus t solves

$$(-.58834842 \quad -.39223228 \quad -.58834839 \quad +.39223226) \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix} = 0$$

Thus t_1, t_2, t_3 can be chosen arbitrarily, but

$$t_4 = 3/2 t_1 - t_2 - 3/2 t_3$$

To satisfy the requirement that $t \geq 0$, it is however necessary that

$$t_2 \leq -3/2 (t_1 + t_3)$$

This can be satisfied only if $t = 0$. Thus a discriminating plane cannot be found. Nevertheless for comparison purposes, the process will be pursued further.

To get $\begin{pmatrix} p \\ d \end{pmatrix} = VQ^+U^T t$, the equation for Q^+ is

$$Q^+ = \begin{pmatrix} .243384 & 0 & 0 & 0 \\ 0 & .2773501 & 0 & 0 \\ 0 & 0 & 2.9053003 & 0 \end{pmatrix}$$

so that

$$VQ^+ = \begin{bmatrix} .1508635 & .1961161 & .9885546 & 0 \\ .1508635 & -.1961161 & .9885546 & 0 \\ -.1171162 & 6.019721E-9 & 2.5468191 & 0 \end{bmatrix}$$

$$VQ^+U^T = \begin{bmatrix} .4230773 & .6153842 & -.5769234 & -.384615 \\ .5769309 & .384615 & -.4230773 & -.6153842 \\ 1.5 & 1 & -1.5 & -1 \end{bmatrix}$$

or more exactly

$$VQ^+U^T = 1/26 \begin{bmatrix} 11 & 16 & -15 & -10 \\ 15 & 10 & -11 & -16 \\ 39 & 26 & -39 & -26 \end{bmatrix}$$

so that

$$\begin{bmatrix} p_1 \\ p_2 \\ d \end{bmatrix} = VQ^+U^T \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix}$$

$$= t_1 \begin{pmatrix} 1 \\ 1.5 \\ 0 \end{pmatrix} + t_2 \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} + t_3 \begin{pmatrix} 0 \\ -0.5 \\ 0 \end{pmatrix}$$

which is zero since as noted earlier, $t = 0$.

Reducing Computational Size of Discriminating Plane Method

Typically n is much bigger than m . For example there might be $n = 1000$ points in an $m = 10$ dimensional space.

Accordingly U may be very large: $n \times n$ and V very small: $m \times m$. The matrix Q is $n \times m$ with no more than m non-zero diagonal entries. Hence I_1 is $n \times n$ with at least $n-m$ non-zero diagonal entries and I_2 is $n \times n$ with at most m non-zero diagonal entries.

By suppressing zero rows or columns, the problem $I_1 U^T t = 0, t \geq 0$ involves an $(n-m) \times n$ matrix, having 90,000 entries, which is very large. But the problem $U I_2 t \geq 0$ involves an $n \times m$ matrix, of 100 entries which is much smaller.

Thus the latter problem is much more tractable with respect to size. However, it entails two difficulties, first it is a non-standard linear programming problem and second, getting to it requires - it would seem - computation of the very large $n \times n$ matrix U .

It turns out that both difficulties can be overcome. First U need not be computed explicitly. The singular decomposition representation of Z as UQV where $UU^T = I$ and $VV^T = I$ implies that

$$Z^T Z = V^T Q^T Q V$$

Thus the singular decomposition calculation applied to the small $m \times m$ matrix $Z^T Z$ can supply V and $Q^T Q$. Since the entries in Q can be assumed to be non negative, this also supplies Q . But then

$$ZV = UQ$$

$$ZVQ^+ = UD$$

which is UI_2 . Thus the matrix UD which is no bigger than the $n \times m$ matrix Z can be gotten without computing the very large $n \times n$ matrix U . The non standard linear programming problem

$$UI_2 c \geq 0$$

$$1^T UI_2 c = 1$$

is solved next using an approach based on the usual simplex algorithm.

Note that once c is gotten, it can be transformed to q by

$$q = VQ^+ U^T t = VQ^+ U^T UI_2 c$$

$$= VQ^+ I_2 c$$

$$= VQ^+ c$$

which, again, does not require explicit calculation of U .

1.2.5 The Relaxed Discriminant Method

The method described above for construction of a discriminating plane will now be improved upon. When a discriminating plane exists, so that there is a gap between the two sets, the new method produces a pair of parallel planes as far apart as possible, for which the entire gap between them separates the sets. When a discriminating plane does not exist, since the sets overlap, the new method again seeks a pair of parallel planes which are as close to one another as possible, and for which one set is on one side of one plane and the other set is on the other side of the other plane.

The new method, called the Relaxed Discriminant Method is advantageous since it is always able, in theory, to solve any problem and provides additional information where the prior, parent method for discrimination also applies.

Suppose that sets X_1 and X_2 interpenetrate in the sense that there is no plane that lies between the two sets. Consider then the easier task of finding a plane having set X_1 , on one side, and a parallel plane having set X_2 on the other side

$$\begin{aligned} X_1 p &\geq d_1 \\ X_2 p &\leq d_2 \end{aligned}$$

where $d_2 \geq d_1$. Thus there can be points, say x , which satisfy both inequalities

$$d_2 \geq xp \geq d_1.$$

By making d_2 and d_1 as close together as possible, a band is determined which discriminates between the two sets in the sense that all points on one side of the band belong to X_1 , and all points on the other side belong to X_2 .

Thus the band satisfies a relaxed discriminant problem. But it serves other important purposes. First it identifies the troublesome points, being the points within the band. These may be found to be suspect for some reason. Or the points may be found to have sufficient observational error that the band can be collapsed to a plane. If the validity of the points is, upon examination, not impugned, then three approaches are available. One approach is to say that within the band, the sets cannot be discriminated - which may be acceptable if the band is narrow. A second approach is to introduce probabilities so that all sets are assumed to interpenetrate and all that can be said is the probability that a point belongs to any particular set. The third approach is to use a nonlinear discriminant surface, which would fit within the band separating the members of the sets within the band. It would seem that constructing such a surface would be helped by knowledge of the band and the difficult points within the band.

Note that the case $d_1 \supseteq d_2$ is also of considerable interest since a band is thereby defined which contains points of neither set. The plane bisecting the band discriminates the two sets, and would not have any marginal points which are on the border line between the two sets. The band, however, also indicates a region in which there is no data, and might with more data be found to be assignable to the two sets in some manner.

The problem to be solved is therefore to find p , d_1 and d_2 such that

$$\begin{aligned} x_1 & p \supseteq d_1 \\ x_2 & p \leq d_2 \\ d_2 - d_1 & \supseteq 0 \end{aligned}$$

For the case of interpenetrating sets $d_2 - d_1$ cannot be less than some positive quantity; it is desirable to minimize $d_2 - d_1$.

The above problem for interpenetrating sets can be written in matrix form as maximizing $d_1 - d_2$ subject to

$$\begin{pmatrix} x_1 & -1 & 0 \\ x_2 & 0 & +1 \\ 0 & -1 & +1 \end{pmatrix} \begin{pmatrix} p \\ d_1 \\ d_2 \end{pmatrix} \geq 0$$

or

$$Zq \geq 0$$

The solution is given by $q = Z^+t$ provided that

$$t \geq 0$$

and

$$(ZZ^+ - I) t = 0$$

The latter equation reduces as before to

$$I_1 U^T t = 0$$

where U is part of the singular value decomposition of Z , that is $Z = UQV^T$ and I_1 is a diagonal matrix whose ii entry is 0 if $q_{ii} \neq 0$ and +1 if $q_{ii} = 0$.

If there are n rows in X_1 and X_2 combined, and they have m columns each, then U is an $(n+1) \times (n+1)$ matrix and Q is $(n+1) \times (m+2)$.

If r is the rank of Q then I_1 has only $n+1-r$ non zero rows. Thus $I_1 U^T$ can be written as an $(n+1-r) \times (n+1)$ matrix.

The problem is now in standard linear programming form: find t such that

$$t \geq 0$$

$$I_1 U^T t = 0$$

$$\text{maximize } -t_{n+1}$$

If the constraints however are satisfied by a vector \hat{t} , then they are also satisfied by any positive multiple of \hat{t} . The objective function would drive that multiple to 0.

This unsatisfactory situation also can be seen in the problem as originally stated, by letting $p = 0$, $d_1 = 0$, $d_2 = 0$. Clearly it arises because no constraint was imposed on the length of p , such as the classical constraint that $\|p\| = 1$. Such a constraint would introduce a nonlinear aspect and so is to be avoided. Instead the linear constraint

$$\sum_{i=1}^n t_i \geq 1$$

will be used. The objective function should drive any solution to the $\sum t_i = 1$ plane, since if t is a solution, not on the planes then $k \hat{t}$, $k = 1 / \sum \hat{t}_i$ is also a solution has smaller t_{n+1} and is on the plane.

The new constraint specifies that the distances of each point from its bounding plane add to at least one; the distances being defined so as to be always positive.

Consider now the problem when the sets are separated, and as wide a band as possible is sought between them. In algebraic terms the problem is to find p , d_1 , d_2 so that

$$\begin{pmatrix} X_1 & -1 & 0 & p \\ -X_2 & 0 & +1 & d_1 \\ 0 & 1 & -1 & d_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ \cdot \\ \cdot \\ t_{n+1} \end{pmatrix} \geq 0$$

and $t_5 = d_1 - d_2$ is maximized.

Converted to standard linear programming form, the problem is to find t such that

$$t \geq 0$$

$$I_1 U^T t = 0$$

$$\text{Maximize } t_{n+1}$$

Here again, if the constraints are satisfied by \hat{t} , they are also satisfied by any positive multiple of \hat{t} , and the objective function drives that multiple to infinity. The reason is that the problem has not been normalized. To do so here the additional constraint

$$\sum_{i=1}^n t_i \leq 1$$

$$i=1$$

is imposed.

The intersection and the separation problems can now be combined as:

Find p, d_1, d_2 such that

$$\begin{pmatrix} X_1 & -1 & 0 \\ -X_2 & 0 & +1 \\ 0 & +1 & \pm 1 \end{pmatrix} \begin{pmatrix} p \\ d_1 \\ d_2 \end{pmatrix} - \begin{pmatrix} t_1 \\ t_2 \\ . \\ . \\ . \\ t_{n+1} \end{pmatrix} \geq 0$$

$$\pm \sum_i t_i \geq \pm 1$$

$$\text{maximize } t_{n+1}$$

or, in standard form: Find t such that

$$t \geq 0$$

$$I_1 U^T t = 0$$

$$\pm \sum_i t_i \geq \pm 1$$

$$\text{Maximize } \mp t_{n+1}$$

where the upper signs are for the intersection problem and the lower for the separation problem. Here UQV^T is the singular value decomposition of

$$Z = \begin{pmatrix} X_1 & -1 & 0 \\ -X_2 & 0 & +1 \\ 0 & \mp 1 & \pm 1 \end{pmatrix}$$

and I_1 is a square diagonal matrix whose ii entry is 0 if $q_{1i} \neq 0$ and is +1 if $q_{1i} = 0$.

Since the labeling of sets is arbitrary, it is of interest to determine what happens if the labels on sets 1 and 2 were interchanged.

The matrix equation in geometric variables is

$$\begin{pmatrix} X_2 & -1 & 0 \\ -X_1 & 0 & +0 \\ 0 & \mp 1 & \pm 1 \end{pmatrix} \begin{pmatrix} p \\ d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ \vdots \\ t_{n+1} \end{pmatrix} \geq 0$$

This can be rewritten successively as

$$\begin{pmatrix} -X_2 & -1 & 0 \\ X_1 & 0 & +0 \\ 0 & \mp 1 & \pm 1 \end{pmatrix} \begin{pmatrix} -p \\ d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ \vdots \\ t_{n+1} \end{pmatrix} \geq 0$$

$$\begin{pmatrix} X_1 & +1 & 0 \\ -X_2 & -1 & +1 \\ 0 & \pm 1 & \mp 1 \end{pmatrix} \begin{pmatrix} -p \\ d_2 \\ d_1 \end{pmatrix} = \begin{pmatrix} t_{n1+1} \\ . \\ . \\ . \\ t_{n1+n2} \\ t_1 \\ . \\ . \\ . \\ t_{n1} \\ t_{n+1} \end{pmatrix} \geq 0$$

$$\begin{pmatrix} X_1 & +1 & 0 \\ -X_2 & -1 & +1 \\ 0 & \mp 1 & \pm 1 \end{pmatrix} \begin{pmatrix} -p \\ -d_2 \\ -d_1 \end{pmatrix} = \begin{pmatrix} t_{n1+1} \\ . \\ . \\ . \\ t_{n1+n2} \\ t_1 \\ . \\ . \\ . \\ t_{n1} \\ t_{n+1} \end{pmatrix} \geq 0$$

The normalizing constraint

$$\pm \sum t_i \geq +1$$

is unchanged as is the objective function.

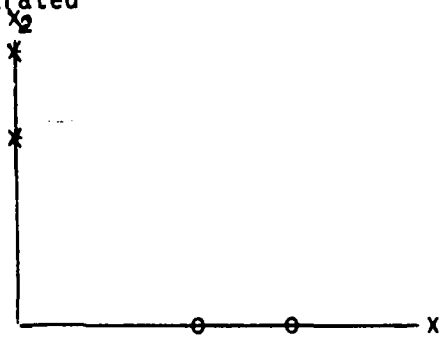
Thus, the solution of the problem for sets X_2, X_1 is the negative of the solution for sets X_1, X_2 . Thus, the same discriminating planes result no matter which labeling is used.

Example 1: The two sets are separated

$$n_1=2, n_2=2, n=4, m=2$$

$$X_1 = \begin{bmatrix} 2 & 0 \\ 3 & 0 \end{bmatrix} \text{ marked with } o$$

$$X_2 = \begin{bmatrix} 0 & 2 \\ 0 & 3 \end{bmatrix} \text{ marked with } x$$



It is required to solve

$$\begin{pmatrix} 2 & 0 & -1 & 0 \\ 3 & 0 & -1 & 0 \\ 2 & 0 & 1 & 1 \\ 0 & -3 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{pmatrix} \geq 0$$

Solving the first four equations gives

$$\begin{pmatrix} p_1 \\ p_2 \\ d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} -1 & 1 & & \\ & & 1 & -1 \\ -3 & 2 & & \\ & & & 3 & -2 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix}$$

so that

$$t_5 = d_1 - d_2 = -3(t_1 + t_3) + 2(t_2 + t_4)$$

The constraint

$$1 = t_1 + t_2 + t_3 + t_4$$

implies that

$$-(t_1+t_3)=(t_2+t_4)-1$$

so that

$$\begin{aligned} t_5 &= 3[-1+(t_2+t_4)]+2(t_2+t_4) \\ &= 5(t_2+t_4)-3 \end{aligned}$$

and $t_5 \geq 0$ requires that $t_2+t_4 \geq 3/5$

Since d_1-d_2 , which is t_5 , is to be maximized, it follows that t_2+t_4 is to be maximized. Since the t 's are non-negative, the optimum choice is

$$t_1=t_3=0 \qquad t_2+t_4=1$$

This gives

$$p_1 = t_2$$

$$p_2 = -t_4 = t_2-1$$

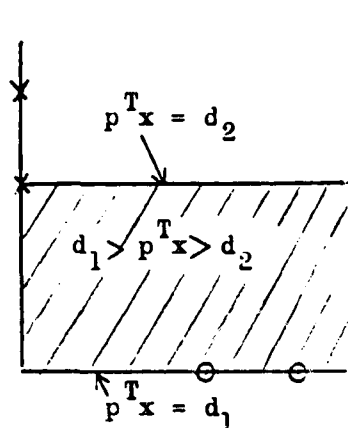
$$d_1 = 2t_2$$

$$d_2 = -2t_4 = 2t_2-2$$

and

$$d_1-d_2 = 2$$

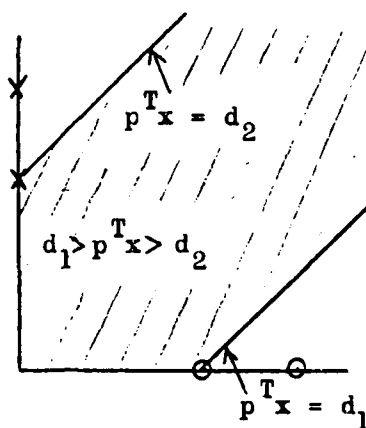
with t_2 undetermined in the range of $0 \leq t_2 \leq 1$. The following shows the situation for $t_2=0, 1/2, 1$.



$$t_2 = 0$$

$$p = \begin{pmatrix} 0 \\ -1 \end{pmatrix}, d_1 = 0, d_2 = -2$$

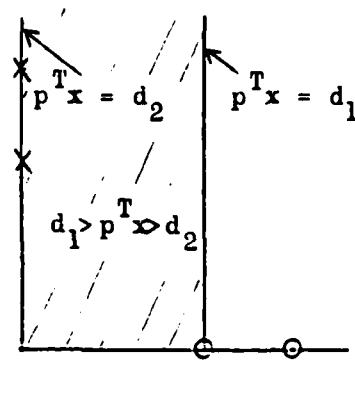
$$d_1 - d_2 = 2$$



$$t_2 = 1/2$$

$$p = \begin{pmatrix} 1/2 \\ -1/2 \end{pmatrix}, d_1 = 1, d_2 = -1$$

$$d_1 - d_2 = 2$$



$$t_2 = 1$$

$$p = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, d_1 = 2, d_2 = 0$$

$$d_1 - d_2 = 2$$

Here the shaded region is prohibited for either set of points. Now consider the inequality constraint

$$1 \geq t_1 + t_2 + t_3 + t_4$$

which says that

$$(t_1 + t_3) \leq -(t_2 + t_4) + 1$$

The constraint $t_5 \geq 0$ implies that

$$(t_1 + t_3) \leq (2/3)(t_2 + t_4)$$

Choosing $t_1 + t_3 = 0$ does not constrain the choice of $t_2 + t_4$ and benefits the objective. The constraint then on $t_2 + t_4$ is simply

$$0 \leq t_2 + t_4 \leq 1$$

Choosing $t_2 + t_4 = 1$ benefits the objective most.

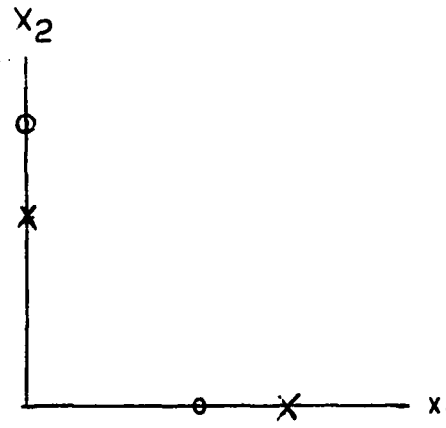
Thus, the inequality constraint $1 \geq t_1 + t_2 + t_3 + t_4$ leads to the same result as the equality constraint $1 = t_1 + t_2 + t_3 + t_4$.

Example 2: The two sets are separated

$$n_1=2, n_2=2, n+4, m=2$$

$$X_1 = \begin{pmatrix} 0 & 2 \\ 3 & 0 \end{pmatrix} \text{ marked with } o$$

$$X_2 = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \text{ marked with } x$$



It is required to solve

$$\begin{pmatrix} 0 & 2 & -1 & 0 \\ 3 & 0 & -1 & 0 \\ -2 & 0 & 0 & 1 \\ 0 & -3 & 0 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{pmatrix} \geq 0$$

Solving the first four equations gives

$$\begin{pmatrix} p_1 \\ p_2 \\ d_1 \\ d_2 \end{pmatrix} = 1/5 \begin{pmatrix} -3 & 3 & 2 & -2 \\ -2 & 2 & 3 & -3 \\ -9 & 4 & 6 & -6 \\ -6 & 6 & 9 & -4 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{pmatrix}$$

so that

$$t_5 = -d_1 + d_2 = (3/5)(t_1 + t_3) + (2/5)(t_2 + t_4)$$

The constraint $t_5 = 0$ is implied by the constraints $t_i \geq 0, i=1, \dots, 4$.

The constraint

$$1 = t_1 + t_2 + t_3 + t_4$$

implies that

$$t_5 = (3/5)[1 - (t_2 + t_4)] + (2/5)(t_2 + t_4)$$

$$= 3/5 - (1/5)(t_2 + t_4)$$

Since $d_1 - d_2$, which is $-t_5$, is to be maximized, $t_2 + t_4$ is to be maximized. Since the t 's are non-negative, the optimum choice is

$$t_1 = t_3 = 0, \quad t_2 + t_4 = 1$$

This gives

$$p_1 = (3/5)t_2 - (2/5)t_4 = t_2 - 2/5$$

$$p_2 = (2/5)t_2 - (3/5)t_4 = t_2 - 3/5$$

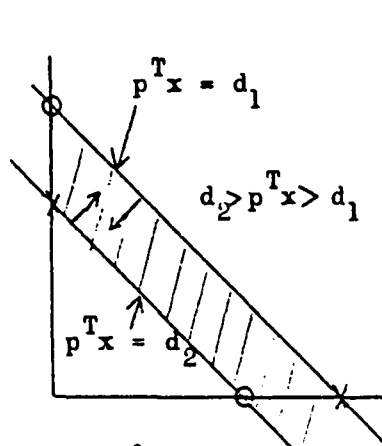
$$d_1 = (4/5)t_2 - (6/5)t_4 = 2t_2 - 6/5$$

$$d_2 = (6/5)t_2 - (4/5)t_4 = t_2 - 4/5$$

and

$$d_1 - d_2 = -2/5$$

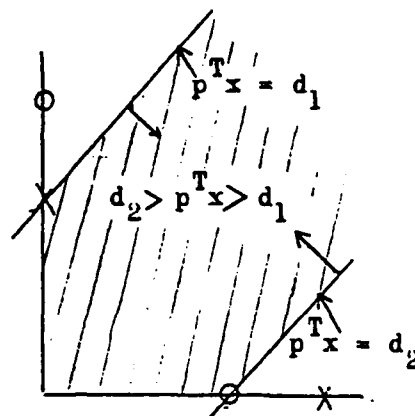
with t_2 undetermined in the range of $0 \leq t_2 \leq 1$. The following shows the situation for $t_2 = 0, 1/2, 1$.



$$t_2 = 0$$

$$p = -\frac{1}{5} \begin{pmatrix} 2 \\ 3 \end{pmatrix}, d_1 = -\frac{6}{5}, d_2 = -\frac{4}{5}$$

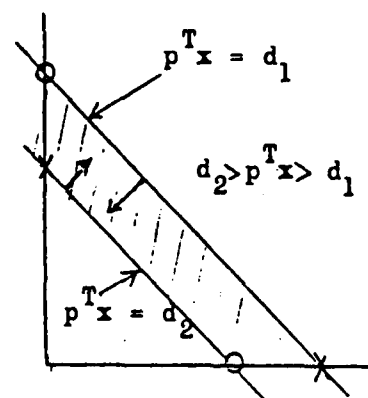
$$d_1 - d_2 = -2/5$$



$$t_2 = 1/2$$

$$p = \frac{1}{10} \begin{pmatrix} 1 \\ -1 \end{pmatrix}, d_1 = -\frac{1}{5}, d_2 = \frac{1}{5}$$

$$d_1 - d_2 = -2/5$$



$$t_2 = 1$$

$$p = \frac{1}{5} \begin{pmatrix} 3 \\ 2 \end{pmatrix}, d_1 = \frac{4}{5}, d_2 = \frac{6}{5}$$

$$d_1 - d_2 = -2/5$$

The shaded region is the overlap region available to both sets.
Consider now the inequality constraint

$$1 \leq t_1 + t_2 + t_3 + t_4$$

The objective

$$-t_5 = -(1/5)(t_1 + t_3) - (2/5)(t_1 + t_2 + t_3 + t_4)$$

is maximized by choosing $t_1 + t_3$ small, as well as $t_1 + t_2 + t_3 + t_4$ small; these are not in conflict. Thus $t_1 = t_3 = 0$ and $t_2 + t_4 = 1$ as was found before.

1.2.6 Cluster Analysis

A cluster might be defined as a set of points such that between any two points x^A, x^B in the cluster there is a sequence of points in the cluster: $x^A, x^1, x^2, \dots, x^I, x^B$ which are close to one another in the sense

$$\begin{aligned} \|x^A - x^1\| &\leq d \\ \|x^i - x^{i+1}\| &\leq d, \quad 1 \leq i \leq I-1 \\ \|x^I - x^B\| &\leq d \end{aligned}$$

for some distance d . Here $\|x\|$ means Euclidean length that is $\|x\| = [\sum x_i^2]^{1/2}$ where the x_i are the components of x .

Which points would be included in the cluster would clearly be dependent on the value of d . For some situations, in which a reasonable value of d may not be clear, it is therefore desirable to define a cluster differently in a more intrinsic, topological manner, independent of the choice of d . Such a definition has a special significance even when suitable d values may be estimated, since the clusters thus defined have an absolute, intrinsic n .

The new definition will involve use of two metrics, the familiar Euclidean metric

$$\|x^A - x^B\| = \left[\sum_{k=1}^n (x_k^A - x_k^B)^2 \right]^{1/2}$$

and a new metric which will be called a Stepping Stone metric.

Between points x^A and x^B , there are many sequences

$$s = [x^A = x^0, x^1, x^2, \dots, x^I, x^{I+1} = x^B]$$

WD-8193-699

EMPIRICAL FLUTTER PREDICTION METHOD(U) GE AIRCRAFT
ENGINES CINCINNATI OH ADVANCED TECHNOLOGY OPERATION
J K CASEY 05 MAR 88 R87AEG AFWAL-TR-87-2087

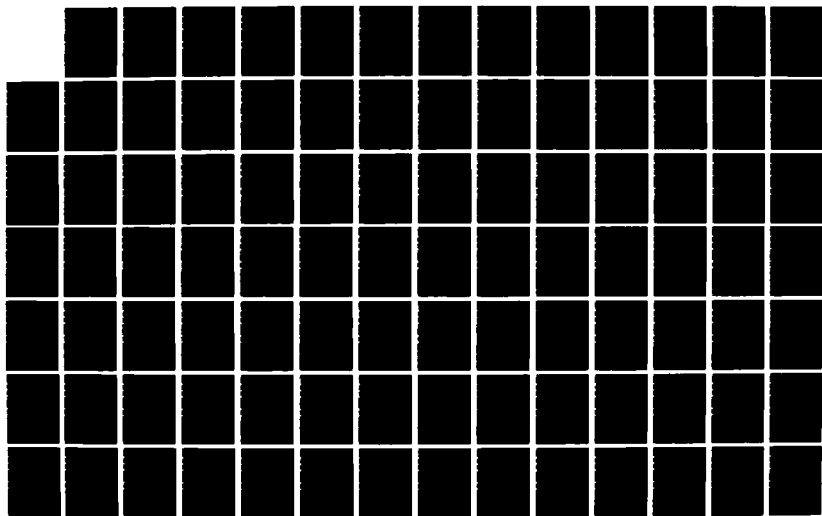
274

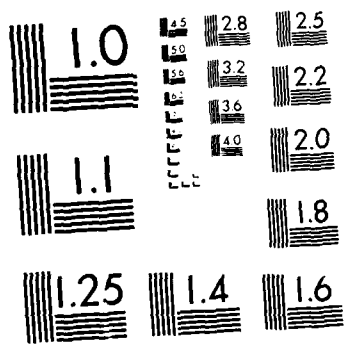
UNCLASSIFIED

F33615-84-C-2457

F/G 20/4

NL





of points. For each sequence, the maximum distance between successive points is

$$d_s = \max_{0 \leq i \leq I} \|x^i - x^{i+1}\|$$

where I depends on the sequence s . The quantity d_s will be called the Stepping Stone distance along the sequence s from x^A to x^B . For the set S of all such sequences, choose the least of the d_s values

$$d = \min_{s \in S} d_s$$

Thus, d is the largest step that one is obligated to take in going from x^A to x^B . The quantity d will be called simply the Stepping Stone distance between points x^A and x^B .

This distance satisfies the three properties of a metric since the Stepping Stone distance is always non-negative, is 0 if $x^B = x^A$, is the same from x^A to x^B as from x^B to x^A , and it satisfies the triangle inequality:

$$d(x^A, x^C) \leq d(x^A, x^B) + d(x^B, x^C)$$

since the path from x^A to x^B to x^C having Stepping Stone distances $d(x^A, x^B)$ and $d(x^B, x^C)$, respectively, is one of the paths considered for $d(x^A, x^C)$ and for this particular path the Stepping Stone distance is

$$\min(d(x^A, x^B), d(x^B, x^C))$$

which is less than or equal to the sum.

The Stepping Stone distance between x^A and x^B is thus the Euclidean distance between two points, say x^I, x^{I+1} in a certain sequence

$$x^A, \dots, x^I, x^{I+1}, \dots, x^B$$

which goes from x^A to x^B . The point x^I is defined as the frontier point for going from x^A to x^B and x^{I+1} is defined as the frontier point for going from x^B .

A cluster is defined as a set of points which includes all their frontier points and cannot be decomposed into smaller clusters. A point which is not a frontier point in going from one cluster to another is called an interior point.

The computation of clusters according to these concepts can best be carried out in terms of two $m \times m$ matrices where m is the number of data points. The i,j entry for one matrix is the Stepping Stone distance from point i to point j . In another matrix, the i,j entry is the frontier point number for going from point i to point j .

The matrix of Stepping Stone distances is initialized to the Euclidean distance between points. The matrix is continually swept, replacing at each step the ij entry by the maximum of the i,k and k,j entries if this produces a smaller ij entry. Thus, if (the biggest step required in going from point i to point k) and (the biggest step required in going from point k to point j) are less than (the biggest step found thus far to be required in going from i to j), then the maximum of the former replaces the latter. The sweeping continues until no more change occurs.

The matrix of frontier points is initialized so that the i,j entry is i if below the diagonal and is j if above the diagonal, the diagonal entries being set to 0 and subsequently ignored. If the ik Stepping Stone distance replaces the ij Stepping Stone distance, then the ik frontier entry replaces the ij entry and the ki entry replaces the ji entry. Conversely, if the kj Stepping Stone entry replaces the ij entry, then the kj frontier entry replaces the ij entry and the jk entry replaces the ji entry.

Membership in the first cluster is determined by picking a point arbitrarily, then including it and all its frontier points in the

cluster, then all their frontier points, etc., and finally all interior points which have frontier points in the cluster. Membership in the next cluster is determined in the same way beginning with any point not already assigned to a cluster.

Certain refinements may be mentioned but will not be discussed in detail. These include: the case in which the ik Stepping Stone distance equals the kj , and is less than the ij Stepping Stone distance, or the case in which both equal the ij Stepping Stone distance; the case in which subclusters exist; the case in which bonds are defined to force two points and their clusters to be combined into the same cluster.

A special point of interest is that this approach to clusters does not require a prior definition of smallness by the user, which can often be a difficult requirement. Instead, the definition of cluster here is more intrinsic.

Another interesting feature of this approach is that two metrics are being simultaneously employed. And the pivotally important frontier points have the property that the two metrics coincide on pairs of frontier points, which are thus intrinsically distinguished points.

In establishing certain properties of clusters generated in this way, some notation will be useful. Let a_1, a_2, \dots be interior points and $\alpha_1, \alpha_2, \dots$ frontier points of a cluster A ; and b_1, b_2, \dots and β_1, β_2, \dots the interior and frontier points of a different cluster B ; etc. The Stepping Stone metric will be assumed in speaking of the distance between points, or the closeness of points.

Frontier points are defined in pairs; one frontier point is said to be the pair of the other. The points of a cluster for which a point α is a frontier point are said to belong to α . Also, the set of points which are frontier points for an interior point a , are said to belong to a . The operator T applied to a set of points produces all the points which are frontier points for at least one of the interior points.

$$T(a) = [\alpha_1, \alpha_2, \dots]$$

The inverse operator T^{-1} applies to frontier points $\alpha_1, \alpha_2, \dots$ and produces all points for which these are frontier points.

$$T^{-1}(\alpha) = [a_1, a_2, \dots]$$

Note that a frontier point is a frontier point for itself. Also, a frontier point α_1 in going from cluster A to cluster B may use α_2 as its frontier in going from cluster A to cluster C.

Points belonging to a frontier point α are no further from one another than α is from its pair. This follows since otherwise one such point would be further from α , and from its pair than is α .

We can show that the frontier point for a cluster which is most distant from its pair is a frontier point for all points in its cluster. Thus, the points of a cluster are no further from one another than this frontier point is from its pair. Hence, there is at least one frontier point which can serve the entire cluster. Such a frontier point will be called a capital frontier point.

To prove this result, suppose the contrary, that α_1 cannot serve the entire cluster. Suppose that the frontier point in question, say α_1 , is a distance d_1 from its pair β_1 , and that the points $T^{-1}(\alpha_1)$ which belong to it do not exhaust the cluster but that there is another frontier point α_2 whose pair β_2 is in the same cluster B as β_1 and that $T^{-1}(\alpha_2)$ has no points in common with $T^{-1}(\alpha_1)$. Suppose that the distance from α_2 to β_2 is d_2 , and that d is the least distance between points of $T^{-1}(\alpha_1)$ and points of $T^{-1}(\alpha_2)$. By hypothesis $d_2 \leq d_1$.

If $d \leq d_1$ and $d_2 < d_1$, then there are points of $T^{-1}(\alpha_1)$ which have a shorter path to B by going through α_2 , which is contrary to assumption. If $d \leq d_1$ and $d_2 = d_1$, then α_1 can serve as frontier for both $T^{-1}(\alpha_1)$ and $T^{-1}(\alpha_2)$ which is also contrary to assumption. Hence, d must exceed d_1 .

But since any frontier point for points of $T^{-1}(\alpha_1)$ is no more than d_1 from its pair, no points belonging to them are more than d_1 apart and so there can be no point belonging to them which is in $T^{-1}(\alpha_2)$. Since the points of a cluster are generated by T or T^{-1} operations, no such operation on a point generated from a , can ever produce a point of $T^{-1}(\alpha_2)$. Thus, the assumption that α_1 and α_2 are in the same cluster is untenable. Thus α_1 can serve the entire cluster; any other frontier point α_2 with its pair in B is thus redundant. The other assertions follow directly.

It follows further that if any point of the cluster is chosen, the operation $T(\alpha)$ will produce a capital frontier point and so $T^{-1}T(\alpha)$ will produce the entire cluster. Were this conclusion not established, the questions could be raised whether the membership in a cluster would be independent of which point was selected first as generator, and also whether one sequence of T, T^{-1} would generate a different cluster than another sequence.

Clusters thus have the intuitive property that the distance within the cluster is less than the distance between clusters, where the latter distance means the distance to the furthest cluster.

Clusters have several different uses in the analysis of large sets of data.

One use is to discover types of points. The points of one type will differ in each variable by a large amount from the points of the other type, the quantity being considerably more than the differences within each type. This usage is sometimes called taxonomic since it discovers natural groupings. It has been used in this way to discover species or subspecies of animals, and to discover different types of voter or consumer requiring different persuasions. This usage of clusters is the one most often addressed in the literature.

In the above use, clusters would be sought with respect to behavior or performance or response variables. Once this were done, corresponding clusters might be sought among descriptive or predictive or causal variables. The underlying idea would be that the former are effects while the latter are causes. Dichotomous differences in effects suggest dichotomous differences in causes. This is the case if the cause effect relation is continuous. But it need not always be the case since some causal factors can produce a discontinuous effect relative to some threshold value, so that a small change in the causal variable can produce a discontinuous jump in a response.

The first sort of usage does not apply to the flutter prediction problem. Here the types of behavior are the different kinds of stability condition, which are known beforehand for the Annular Cascade Data Base. What is of interest, therefore, are the corresponding clusters in the space of predictive variables. But here it is well known that there are transition regions where a small change in these variables can make a large qualitative change in the stability condition. Furthermore, the subset of the Annular Cascade Data Base which, in the interest of computer economy was used, are principally the points in these transition regions. Thus the points selected form clusters which intentionally contain points of different stability conditions.

It might be mentioned parenthetically that an example in jet engine technology of the fruitful use of these two types of cluster application might arise in analysis of repair/operational problems of jet engines. Do these separate into clusters? into syndromes? If so, these clusters would be the behavioral types. Then we would ask whether there were corresponding clusters in the space of engine history/operation. These would be the causal clusters. This correspondence would suggest that an engine in a certain operational cluster will develop a certain syndrome of performance problems.

The cluster concept also has a third, quite different use, which is the use of greater importance for the empirical flutter prediction problem.

Suppose there is no big separation between the predictive variables that correspond to one type of behavior and the predictive variables that correspond to a different type of behavior. If the predictive variables are sufficient to predict behavior, there must be a surface running through them such that points on one side produce one type of behavior, while points on the other side produce a different type of behavior.

Discovery of such a surface is the task of discriminant analysis, which is easier to the extent that the surface is not tightly crowded by points on each side. If it is crowded, then a nonlinear surface may be needed, and it will be necessary to identify where the surface is tightly crowded, that is, where the surface must thread through a cluster having points of two or more different behavioral types.

Thus, the third use of cluster analysis is to locate clusters in the space of predictive values for points of two or more behavioral types.

The task of proposing a nonlinear surface separating the two behavioral types is thereby simplified since the clusters can initially be viewed as single "points." The nonlinear surface should traverse the "points" of mixed behavioral type and separate those of single behavioral type.

Indeed, clusters of a single behavioral type may very possibly be irrelevant to the problem of finding a discriminating plane, provided there are enough clusters of mixed stability type to define the planes.

There is still a fourth use of clusters. This concerns the geometric shape of the cluster.

Consider the center of gravity of a cluster, that is the point whose coordinates are the average of the corresponding coordinates of the members of the clusters.

If the center of gravity is inside the cluster, this suggests the cluster is convex; if outside the cluster, concavity is indicated.

The same question can, of course, be asked relative to convex hulls of the cluster. And the answers may be different. For example, if the points of the cluster were distributed over a half circle, the answers would be different; while if they were distributed over a circle, the answers would be the same.

2.0 GALACTIC Applied To Annular Cascade Data Base

GALACTIC was applied to 891 test points selected from the much larger Annular Cascade Data Base.

These points were selected because they were in the transition regions between stability regions. This selection served to reduce the total number of points required and so to minimize computer storage and running time. At the same time, this selection excluded most "easy points," making it difficult to find planes that separated any two stability regions since the plane was constrained to lie between points close on either side.

While the Annular Cascade Data Base contains a great deal of information about each test point, production engine test points would have much less information. Accordingly, it would be impractical to use more information from the Annular Cascade Data Base than could be expected from production engine test points. For this reason, GALACTIC used only the following nine items of aeromechanical data for each test point:

<u>Name</u>	<u>Meaning</u>
PIN	Pressure at inlet
TINLET	Temperature at Inlet
SLDTYA	Solidity
REY	Reynold's number
INC	Leading edge angle of incidence
MLE	Mach number at leading edge
VLE875	Velocity at a point on the leading edge distant from the blade root by 87.5% of the span
REDV1	Reduced velocity - flexure
REDV2	Reduced velocity - torsion

The 891 test points populated 14 stability regions:

Stability <u>Code</u>	Number <u>of Points</u>	<u>Stability Region</u>
00	209	Stall; stable
01	137	Stall; flexural flutter
02	56	Stall; torsional flutter
03	22	Stall; flexural and torsional flutter
04	15	Stall; 2nd flexural flutter
05	2	Stall; 1 flex., 1 tors., 2 flex. flutter
06	1	Stall; 1 flex., 2 flex. flutter
07	31	Stall; separated flow
10	201	Interior, stable
20	73	Choke; stable
21	126	Choke; flexural flutter
22	13	Choke; torsional flutter
23	4	Choke; flex. and torsional flutter
27	<u>1</u>	Choke; separated flow
	891	

These may be combined as:

376	Flutter (9 regions)
483	Stable (3 regions)
<u>32</u>	Separated Flow (2 regions)
891	

or as

473	Stall (8 regions)
217	Choke (5 regions)
<u>201</u>	Interior (1 region)
891	

The 14 stability regions produce

$$13 + 12 + \dots + 1 = 91$$

pairs of stability regions and therefore require an equal number of hyperplanes to separate them - or more correctly - pairs of hyperplanes since the Relaxed Discriminant Method is used.

The 91 hyperplanes were produced over a 23-day period and required 31 computer runs and up to 10 hours VAX4 processor time per run.

2.1 Intersecting Pairs of Stability Regions

For 21 of 91 pairs of stability regions, the Discriminant Feasibility Method found that the regions intersect one another (i.e. the convex hulls intersect). The Relaxed Discriminant method proceeds to find pairs of hyperplanes which in all these cases show an overlap, as expected. The overlaps are generally not large; their width may be compared to one another and to unity since all variables had been normalized to have a range of from 0 to 1. The following provides details on these 21 pairs of stability regions: overlap width, the number of points in each pair of stability regions, the percents of these points which are in the overlap region, which are outside the overlap region on the correct or on the incorrect side.

Stall, Stall Pairs (9 out of 28 pairs = 32%)

<u>Stability Code</u>	<u>Stability Code</u>	<u>Overlap Width</u>	<u>% Correct</u>	<u>% Incorrect</u>	<u>% Overlap</u>	<u>Number of Points</u>
00	01	0.023	40	8	53	346
00	02	0.260	45	0.4	55	265
00	03	0.137	12	0	88	231
00	04	0.043	78	0.4	21	224
00	07	0.104	45	2	54	240
01	02	0.054	23	0	77	193
01	03	0.0002	83	10	7	159
01	07	0.004	61	15	23	168
02	03	0.002	78	8	14	78

Choke, Choke Pairs (2 out of 10 = 20%)

20	21	0.030	62	5	33	199
21	22	0.002	58	13	29	139

Stall, Choke Pairs (1 out of 40 = 2.5%)

02	22	0.017	48	3	49	69
----	----	-------	----	---	----	----

Interior, Stall Pairs (6 out of 8 = 75%)

10	00	0.792	0.7	0.2	99	410
10	01	0.032	66	4	30	338
10	02	0.002	93	3	4	257
10	03	0.00005	96	2	2	223
10	04	0.230	50	0.5	50	216
10	07	0.027	84	2	14	232

Interior, Choke Pairs (3 out of 5 = 60%)

<u>Stability</u> <u>Code</u>	<u>Stability</u> <u>Code</u>	<u>Overlap</u> <u>Width</u>	<u>%</u> <u>Correct</u>	<u>%</u> <u>Incorrect</u>	<u>%</u> <u>Overlap</u>	<u>Number</u> <u>of Points</u>
10	20	0.437	4	0.7	95	274
10	21	0.327	5	0.6	94	327
10	22	0.008	77	3	20	214

Several observations can now be made, based largely on the above details.

- o The interior region showed the most intersections whether with stall or with choke regions.
- o The fewest intersecting pairs occurred between the choke and stall regions.
- o Even for the 21 pairs of intersecting stability regions, there were relatively few test points on the incorrect side of the hyperplanes, and the pairs with the most incorrect points have especially narrow overlap regions. Thus, the incorrect points may have been only slightly in error.
- o Most overlaps have narrow width, suggesting that a slightly nonlinear surface might serve to discriminate between the two stability regions.
- o Substantial numbers of points fall into many of the overlap zones, making the hyperplanes in such cases ambiguous and of little help in discriminating between the stability regions they were to separate.
- o The Discriminant Feasibility Method is virtually foolproof in declaring that two regions intersect since it constructs a common point. The conclusion that two regions are disjoint, that is, that a common point could not be found, is not as foolproof.

- o The Relaxed Discriminant Method is seen to work consistently with the Discriminant Feasibility Method in that, in all 21 cases, it produced hyperplanes with overlaps.

2.2 Disjoint Pairs of Stability Regions

For 70 of 91 pairs of stability regions the Discriminant Feasibility Method found that the regions are disjoint (i.e. the convex hulls do not intersect). This conclusion is not, however, foolproof, since it is based on the negative result that a point common to the two regions could not be found.

However, in all but one case the conclusion was confirmed by the Relaxed Discriminant Method finding discriminating pairs of hyperplanes. Usually, the pairs of planes have a gap, as is expected. But in some cases the iteration counter ran out on the gap case, and the overlap case was tried and resulted in a slight overlap (say less than 0.01), or perhaps a negative overlap. In a few cases, neither the gap nor the overlap case was successfully concluded before the iteration counter was exhausted and the latest hyperplane pair was accepted with, again, no more than a slight overlap.

In three cases there was an important difference between the two methods, namely for regions (00,20), (00,21) and (10,23). Here the Discriminant Feasibility Method declared that the two regions are disjoint, but the Relaxed Discriminant Method was unable to successfully conclude either the gap or overlap case. When the iteration counter ran out, the hyperplane pair had, respectively, an overlap of 1.31 with 22 percent of the points in the wrong region, a gap of 0.002 but 39 percent in the wrong region, and an overlap of 0.281 with 100 percent of the points in the overlap zone.

Two possible explanations are either that the Discriminant Feasibility Method failing to construct a point common to both regions declared

disjointness prematurely, or that the Relaxed Discriminant Method terminated prematurely. Which explanation is correct has not been ascertained.

2.3 Conclusions

The above discussion accounts for all but a very few scattered instances of points falling into overlap regions and, therefore, not being discriminated by the hyperplane pairs - and these points may have been borderline.

For 23 percent of the pairs of stability regions (21 of 91), the stability regions appear to intersect and to require a nonlinear surface to fully discriminate between the regions.

For 74 percent of the pairs of stability regions, a linear discriminating surface, that is a hyperplane, can discriminate between the two regions.

For three cases, 3 percent, it is not clear to which category they belong.

The voting procedure in EFAGHY can compensate to some extent for ambiguous hyperplane pairs, that is pairs with an appreciable overlap zone, since even though the correct stability region may lack some votes due to ambiguous hyperplane pairs, it may nonetheless receive more votes than any other region.

3.0 EFAGHY Computer Program

EFAGHY is an acronym for "Empirical Flutter Analysis by GALACTIC Hyperplanes." This is a Fortran77 program which has been running on the VAX4 computer and is designed to apply a file of hyperplanes produced by GALACTIC to files of test points so as to determine in which stability region each test point is located.

3.1 Hyperplane File from GALACTIC

The file, called FILEHO within EFAGHY, was produced by GALACTIC and has the format, for each hyperplane:

Stability code for 1st stability region
Stability code for 2nd stability region
C' Left hand side constant for 1st stability region
C" Left hand side constant for 2nd stability region
C₁ Right hand side coefficient for 1st variable
C₂ Right hand side coefficient for 2nd variable
.
.
.
.

The pair of parallel hyperplanes have the equations

$$c' = c_1 x_1 + c_2 x_2 + \dots$$

and

$$c'' = c_1 x_1 + c_2 x_2 + \dots$$

For a given test point, the right hand side is computed for the values x_i which occur at the test point.

If c' is less than or equal the right-hand side, the first stability region claims the point while if c'' exceeds or equals

the right-hand side, the second stability region claims the point.

If C' is less than C'' the regions overlap and both regions can claim the point.

If C' exceeds C'' there is a gap between the planes so that a test point which falls in the gap is not clearly claimed by either region.

Note that there is a coefficient for each of the words in the record for each test point. In the application of GALACTIC to the Annular Cascade Data Base, there are 20 words for each data point. The same 20 word format is used for test points employed by GALACTIC to build the hyperplanes as well as test points to which EFAGHY applies the hyperplanes. The words which are not relevant are multiplied by a zero coefficient.

3.2 Test Point Files to which Hyperplanes are Applied

EFAGHY is presently dimensioned to accept up to 14 files of test points. The words per test point record is also input, but must be the same as was used in GALACTIC, in which 20 was used.

3.3 Input to EFAGHY

In addition to the number and names of files of test points and the length of their records and the name of the hyperplane file, EFAGHY asks the name for a file on which to write its output. If the response is null, then the output is printed on the user's terminal device.

Other input allows the input data points to be perturbed a fractional amount both plus and minus in any variable. The amount may be different for each variable.

It is also possible to confine the application of EFAGHY to certain hyperplanes, which must be named by the number in sequence in which they appear on the hyperplane file.

Finally, the output volume is controlled by a 7-digit number IOUTPUT consisting of 0's and 1's. The 0 calls for suppression of an output, and 1 calls for display of output. The various possible output can best be explained in the course of describing the program capability.

3.4 Voting by Hyperplanes

As described earlier there are 14 possible stability regions considered currently by GALACTIC. Between each pair of regions GALACTIC seeks to put a pair of parallel hyperplanes. As the number of pairs among 14 things is 91, there are 91 pairs of hyperplanes. Each pair is intended to separate one stability region from a second stability region.

For a given test point and a given hyperplane, the test point will be situated in one of the three domains into which the pair of hyperplanes divides all of space.

If the point is not situated between the pair of hyperplanes, the point occupies the same half space which contains one of the two stability regions. Thus this pair of hyperplanes indicates that the point has met a necessary (not sufficient) condition for being in one stability region and not being in the other. In this case, the pair of hyperplanes applied to the test point casts a yes vote for one stability region and a no vote for the other.

If, on the other hand, the test point is situated between the pair of hyperplanes, the situation is ambiguous. In case of overlap, it is possible to cast a vote for both, that is, to cast a fractional vote for each region depending on how deep into each region the point is, being 1 or yes for a stability region if on the plane bounding the half space belonging to that region and -1 or no if on the other plane. Thus the vote for each region varies linearly from +1 to -1.

In the case of a gap, the situation is also ambiguous. As with overlap, it is possible to cast a fractional vote for each stability region. If the point is virtually at one plane, a + 1 vote is cast for the stability region it bounds and -1 for the other region. The votes can vary linearly across the gap.

Now each stability region is voted on by 13 hyperplanes. Thus it can get a maximum of 13 yes votes. Since for each such vote, a different one of the other 13 stability regions receives a no vote, it follows that at most one stability region can receive 13 yes votes.

Other stability regions will receive some yes votes. But these should not be regarded as incorrect votes, since a yes vote simply means that the test point satisfies one of the 13 necessary conditions required of points in a particular stability region. A given test point will partially fulfill the requirements for several different stability regions, but it can fulfill all conditions for only one stability region.

Ideally it would be sufficient simply to count the yes votes to determine into which stability region the hyperplanes classified a point. Were there no ambiguous regions between the planes, there would be 91 yes and 91 no votes distributed among 14 stability regions, so that the average number of yes votes would be 6.5 per stability region. Because of the occurrence of ambiguous overlap or gap regions, there can be fewer yes votes. "Y" vote is the term that will be used for the

simple procedure of assigning a point to the stability region which has received the most yes votes, that is not counting no votes, overlap or gap votes.

A somewhat more complicated voting procedure is to give each stability region credit for any overlap or gap region adjacent to a half space belonging to the stability region. No votes are subtracted. This combined vote will be termed a "C" vote.

Another way of combining votes is the weighted vote, or "W" vote. This adds the yes votes, subtracts the no votes and treats the overlap or gap cases as follows. For these cases, the adjacent stability region gets a 1 vote if the test point is on the plane which bounds it, but the vote declines linearly as the test point is further across the overlap region, reaching 0 on the far plane.

A fourth way is the balanced vote, or "B" vote. This is similar to the W vote except that the vote goes from +1 to -1 in the overlap or gap cases rather than from +1 to 0.

A fifth, and final way is the probabilistic or P vote. This is like the B vote except that each vote is multiplied by the probability of an accurate vote for the stability region and by the hyperplane. This is gotten by first applying all hyperplanes to all data base points used to generate the hyperplanes and counting how many points which were in the stability region were correctly identified. Since in some cases there were very few points in a stability region, this observed success ratio was replaced by the lower 50 percent confidence bound for a binomial distribution. If there are many points the confidence band is narrow, while if there are few points it is broad. The final vote for a stability region is then normalized by dividing by the sum of the probabilities used in its calculation.

3.5 Comparison of Vote Count Efficiency

The five different ways of counting votes were in fact developed in the order presented above, by evaluation against the test points from the Annular Cascade Data Base which had been used in producing the hyperplanes.

This information is available from EFAGHY in considerable detail. If the 7th digit of the output control IOUTPUT is 1 rather than 0, then the user is asked to enter up to 20 test points for the following output, the test points being numbered by the order in which they are encountered by EFAGHY. The output has on the left margin the hyperplane number (1 through 91) and the stability code for the first stability region; while on the right margin is the stability code for the second stability region. Between these are 5 columns. In the second and fourth columns are the lesser and the greater of the constants for the two stability regions. Bracketing these two constants are parentheses, like (), in case the stability regions overlap, or parentheses, like) (in case there is a gap between the two stability regions. The actual right-hand side value for the test point in question and the hyperplane is shown in the first, third or fifth columns depending on whether it is less than the two constants, whether it lies between them or whether it exceeds them. If the right-hand side is in the left, or first, column the hyperplanes vote that the point is in the stability region listed on the left margin; conversely for the right.

This highly detailed output allows the user to examine how each individual vote was cast. It allows us to examine why a test point known to be in a certain stability region is not getting the votes it deserves; and why some other stability region is getting more votes than it deserves. Perhaps one or another hyperplane is repeatedly casting wrong votes. Such a hyperplane may need to be recalculated using nonlinear combination variables.

The success for each hyperplane is summarized in a table entitled "Discrimination Success by Hyperplane," which is printed if the sixth

digit of IOUTPUT is 1. This shows for each hyperplane and for both its stability regions the number of yes, no, overlap and gap votes for each point known to be in either region.

If the third digit of the output control IOUTPUT is 1 rather than 0 then a table is printed out for each test point to which the hyperplanes are applied. The table has a row for each stability region and shows the number of yes, no, overlap and gap votes the greatest fractional penetration into a gap or overlap region, as well as the count of C votes, W votes, B votes, P votes. An asterisk marks the maximum vote of each of the five types.

Summary information for all files of test points is provided if the 4th digit of OUTPUT is 1. First there is a separate table entitled "Placement of Identified Sets," for each file, showing for each of the five types of votes how many times the correct stability region, when it was known, was in first place without ties, in the first place with ties, in the 2nd, 3rd place. Another table entitled "Discrimination Success by Set, Using B Votes" shows for each stability region the number of test points for which the stability region is known and there were 0, 1, 2,...stability regions with more B votes. Another table presents the same information but for P votes. This information is summarized still further in a tally of the number of test points for which the P and B votes agree in giving the most votes to the correct stability region, when one does, but not the other, etc.

3.6 Mixed Voting Strategies

Application of the hyperplanes to the test points from the Annular Cascade Data Base from which it was constructed revealed that the P vote and the B vote were more successful than the other three voting procedures in identifying correctly the stability region in which the test points were located.

When both the B vote and the P vote identified the same stability region there was a still higher success rate.

Thus it seemed that in case the stability region was not known, the stability region should be chosen for which both the B vote and P vote agreed. But if they did not agree, what was to be done? Various strategies were evaluated. Here B_1 , means the set with most B votes, B_2 the set with second most B votes. Similarly for P.

Choose set with $B_1 = P_1$, else no choice
Choose set with B_1
Choose set with P_1
Choose set with $B_1 = P_1$, else with $B_2 = P_2$, else no choice
Choose set with $B_1 = P_1$, else $B_2 = P_2$, else $B_2 = P_1$, else
 $B_1 = P_2$, else no choice
Choose set with $B_1 = P_1$, else $B_2 = P_1$, else $B_1 = P_2$, else
 $B_2 = P_2$, else P_1
Choose set with $B_1 = P_1$, else $B_1 = P_2$, else $B_2 = P_1$, else
 $B_2 = P_2$ else P_1
Choose set with $B_1 = P_1$, else $B_1 = P_2$, else no choice
Choose set with $B_1 = P_1$, else $B_1 = P_2$, else B_1 .

For each strategy is supplied a table showing the number of test points for each chosen stability region and for each actual stability region. For perfect success, all off-diagonal entries would be zero. In addition, the probability of success is shown for each chosen stability region, and for all chosen stability regions. This is computed from the observed success rate converted by binomial theory to a 50 percent lower confidence level. The latter recognizes the uncertainty arising from small samples. This probability information by stability region is intended to detect whether some stability regions, when chosen by a certain strategy, deserve more confidence than others.

This information is presented if the sixth digit of IOUTPUT is 1.

3.7 Summary of Output Options

As mentioned earlier, one of the input quantities is a seven-digit word, IOUTPUT, whose digits are 0 or 1, where 0 indicates suppression of output and 1 indicates display of output.

As a memory aid, EFAGHY reminds the user of the function of each digit by use of a three-letter code, for which the details have been supplied in the foregoing text.

The seven codes and their abbreviated meaning are as follows:

<u>Digit</u>	<u>Code</u>	<u>Principal Output</u>
1	HYP	Equations of the hyperplanes
2	PTS	Record (20 words) for each data point
3	VOT	Y, C, W, B, P votes for each stability region for each test point
4	ASS	Assessment of voting success per stability region
5	ASP	Assessment of voting success per point
6	ASC	Assessment of voting success per combined vote strategy
7	HPT	Voting per hyperplane for selected points

4.0 EFAGHY Applied to Annular Cascade Data Base

EFAGHY was first applied to the test points from the Annular Cascade Data Base which were used by GALACTIC to generate the 91 hyperplanes. This was done because the stability region to which each of these points belonged was known. This would serve to calibrate the accuracy of EFAGHY, in three respects before it was applied to the Validation Points.

First it would show to what extent the voting by several hyperplanes could compensate for the fact that some hyperplanes were quite unsuccessful in discriminating between its two stability regions, classifying most points in its two stability regions into the ambiguous overlap category. (As noted elsewhere, this deficiency in some hyperplanes may perhaps be remedied by inclusion of nonlinear combination variables).

Second it would allow for the evaluation of the different ways of counting votes and of the mixed voting strategies.

Thirdly the experience with the Data Base would provide a standard of success against which to appraise the application of the hyperplanes to the validation points, whose stability region was not disclosed beforehand.

4.1 Detailed Examination of 26 Points from File 55777B

This was the first file of test points from the Annular Cascade Data Base, for which the stability region was known, and to which the hyperplanes were applied by EFAGHY. This file was chosen simply because it was the smallest.

Using the Y, C, W votes initially and later the B and P votes, we found that about 77 percent of these test points were assigned by the hyperplanes to the stability region to which GALACTIC had been told they belonged. These points included 12 stable points, 2 stall flex flutter and 6 choke flex flutter points.

Upon examination of the aeromechanical data for each of the remaining six points, we found that the stability region assigned by EFAGHY was actually even more correct in a practical sense. In the case of 3 test points the listed stability condition coexisted with the stability condition selected by EFAGHY and was given the second highest number of votes.

In the case of the other 3 test points, the stability region with the most votes prevailed very near to the test point and in the two cases the listed stability condition had the second highest number of votes, while in one case it received 9.2 the fourth highest number of votes with the regions with more votes: 12.2, 11.4, 11.0 all prevailing near the test point.

Thus from the point of view of practical use it may be said that 100 percent rather than 77 percent of the votes were correct.

It may however be asked, how it can happen that the hyperplanes do not score perfectly on reproducing the listed stability conditions since this information was available to GALACTIC when it was constructing the hyperplanes.

One reason is that for some pairs of stability regions, it is not possible to separate one region from the other by use of a (multi-dimensional) plane. In such cases, it may be that separation would be possible using nonlinear combinations of the given input variables. It may be of course that other variables beyond those currently used may be needed.

In such cases GALACTIC will necessarily produce discriminating planes with an overlap region between them and containing test points which the hyperplanes cannot assign to either region. For about 21 hyperplanes, most of the test points which they are intended to separate, fall into the ambiguous overlap region. Such planes may have poor reliability also with respect to orientation.

When used by EFAGHY, such hyperplanes can be doubly misleading since not only do the points in either of the two stability regions receive at best a weak overlap vote, but points which actually are not in either of the two regions may receive a yes vote due to the unreliability of such planes. While the P vote was introduced to limit the impact of such hyperplanes, it cannot compensate for the advantage to a stability region which is runner up not to be burdened with weak, unreliable hyperplanes.

Besides this reason, for some hyperplanes there are test points in the stability regions to be separated, which are placed not in the ambiguous overlap region but actually in the region belonging to the incorrect stability condition. This has occurred in a relatively few cases. The test point data have undergone considerable transformation before it emerges as an equation of a hyperplane pair and thus some points may violate the hyperplanes which they generated. In recognition of this, GALACTIC adjusts the hyperplane position, not the orientation so as to eliminate borderline violations, but limits the amount of adjustment. We need to verify the cause of such violations.

4.2 Detailed Examination of 57 Points from File C2BDY

For these points, 51 percent were assigned by EFAGHY with the listed stability condition. But using the detailed aeromechanical data for each of the missed points, we found that the stability condition chosen by EFAGHY

coexisted at 6 points and that for 4 others the chosen stability region was close. One point was off the stability map and is regarded as somewhat anomalous; accordingly it was simply excluded.

This results in 39 hits in 56 points or 70 percent success rate.

For the remaining 17 points, only 3 are regarded as bad misses since EFAGHY gave the correct answer the third or fourth highest number of votes. Thus no hint of the correct answer was provided in 5 percent of the cases.

4.3 Generic Examination of Data Base

The detailed examination of 85 data points as described above resulted in the fine tuning of the type of voting and of mixed voting strategies which would produce the greatest number of correct choices of stability region by EFAGHY.

The best method seems to be the P vote, that is to choose P_1 , the stability region with the most P votes. Nearly as good is to choose B_1 , the stability region with the most B votes. When these two agree, there is still greater likelihood of a correct answer. However, the P_2 and B_2 stability regions should also be considered since when the P_1 or B_1 choices are incorrect, it is very often the case that the correct, listed stability region was second choice. This was the case in the preceeding detailed examination of 85 data points, where the listed region was sometimes second choice with the most votes going to a coexisting region or a very near region.

For the 891 data points from the Annular Cascade Data Base which were used by GALACTIC to generate the hyperplanes and which included the 85 data points, it was not possible to carry out the detailed examination as was done on the 85 data points. Thus the only question that could be asked is whether the listed stability condition was chosen.

For 84 percent of the 891 data points the listed stability condition was among B_1 , B_2 , P_1 , P_2 .

For 74 percent of the points, B_1 and P_1 agreed and when this occurred 87 percent of the time the choice was correct. If it is acceptable to refrain from a choice when B_1 and P_1 do not agree, a success rate of 87 percent is therefore indicated.

However if it is always required to make a choice, the best strategy seems to be P_1 , which has a 59 percent success rate. The next best strategy seems to be B_1 , which has a success rate of 56 percent.

Detailed examination of the 26 points, from File 55777B resulted in raising the 77 percent success rate for literal correctness to 100 percent for practical correctness, a jump of 23 percent. For the 57 points of File C2BDY raised the 51 percent success rate for literal correctness to 70 percent for practical correctness. Combined, the success rate for literal correctness was 59 percent, whereas detailed examination raised the success rate for practical correctness to 79 percent a jump of 20 points.

For the 891 test points, P_1 choice also gave a success rate of exactly 59 percent for literal correctness. Were a detailed examination made of the aeromechanical situation of the missed points, it would therefore not be surprising if a success rate of 79 percent were achieved for practical correctness.

In view of the fact that only about 74 percent of the hyperplane pairs are fully adequate, it would not be surprising if only 79 percent of the test points were correctly classified. If non-linear terms could raise to near 100 percent the number of discriminating surface pairs, then a comparable improvement in the success rate is anticipated.

5.0 Selection of Validation Points

5.1 Types of Test Points Proposed

The technical proposal for this contract specified that the Validation test points would include data from fan and compressor stages of advanced military engines, commercial engines, and development engines and that the stability regions would include the stall and choke flutter boundaries, the stable operating regions and the stall and choke regions. The number of Validation Points was not specified.

5.2 Actual Test Points Selected

A total of 51 test points were selected. These were distributed among seven sets of test data, as follows

	<u>Stability Code</u>						
	<u>00</u>	<u>01</u>	<u>02</u>	<u>03</u>	<u>07</u>	<u>10</u>	<u>Total</u>
Aeromechanical Compressor/ High Aspect Ratio		2	4				6
Development Fighter Engine Case 1 Fan Rotor 1st Stage	2	5					7
Development Fighter Engine Case 2 Fan Rotor 2nd Stage	1	4					5
Development Fighter Engine Case 3 Compressor Rotor Stage 4	3	3					6
High Bypass Turbofan Engine 20" Simulator	3	2			2		7
Rotating Rig Test		4	4	1		6	15
High Bypass Turbofan Engine	3	2					5
	<u>12</u>	<u>22</u>	<u>8</u>	<u>1</u>	<u>2</u>	<u>6</u>	<u>51</u>

The stability codes at the head of the columns have the meaning:

- 00 Stall; stable
- 01 Stall; flexural flutter
- 02 Stall; torsional flutter
- 03 Stall; flexural and torsional flutter
- 07 Stall; separated flow
- 10 Interior; stable

We had intended to include some choke points from J85 data. However, we found on closer examination of the data, that choke and stall occurs simultaneously, the former at mid span and the latter at tip. Thus clear Validation Points are not available for choke. Accordingly no Validation Points of choke type were included.

5.3 Aeromechanical Information per Test Point

In contrast to the Annular Cascade Data Base, for which the test points were richly instrumented and measured with special care, the Validation Points come from a non-research environment for which less information was available and what information was available had less precision.

The aeromechanical information which was generally available for production engines, to which this contract work is addressed consists of the following:

<u>Word #</u>	<u>Name</u>	<u>Meaning</u>
2	PIN	Pressure at inlet
3	TINLET	Temperature at inlet
6	SLDTYQ	Solidity
9	REY	Reynold's number
10	INC	Leading edge angle of incidence
11	MLE	Mach number at leading edge
14	VLE875	Velocity at a point on the leading edge distant from the blade root by 87.5% of the span
15	REDV1	Reduced velocity - flexure
16	REDV2	Reduced velocity - torsion

The word number denotes where the data occurs in the 20-word record used for each test point; the same format being used both for points from the Annular Cascade Data Base as well as for the Validation Points.

In anticipation of the limited amount of information available for the Validation Points, it was necessary for GALACTIC to use no more than that information from the much richer Annular Cascade Data Base in generating hyperplanes. Otherwise the hyperplanes, though doubtlessly more correct, would have been inapplicable to real world engine data. Indeed this restriction on the aeromechanical data that could be used might be one of the causes of some hyperplane pairs with heavily populated overlap regions wherein discrimination between the two stability regions was not possible.

6.0 Application of EFAGHY to Validation Points

Once GALACTIC had produced 91 pairs of hyperplanes, these were applied to the 891 test points from which the hyperplanes had been constructed, thereby developing the voting system contained in EFAGHY. We found that 59 percent of the test points were correctly identified in a narrow literal sense by the hyperplane/voting system, and there is reason to believe that 79 percent of the test points were correctly identified in a broader practical sense. This is not surprising since only 74 percent (67 of 91) of the hyperplanes were fully adequate, some 21 of the stability region pairs showing intersections which might be avoided by curved discriminating surfaces and 3 requiring improvement.

We further found that while the best two voting approaches were to choose P_1 or B_1 (the top vote getters for P votes or B votes), giving the mentioned 59 percent literal accuracy, if both approaches agreed ($P_1 = B_1$) then there was an 87 percent probability of the choice being correct in a literal sense.

Finally we found it was 84 percent certain that the correct stability region would be P_1 the region with most P votes or P_2 the runner up, or B_1 the region with most B votes, or B_2 the runner up.

EFAGHY was then applied to the 51 Validation Points. The results will be compared with those achieved when EFAGHY was applied to the 891 test points from the Annular Cascade Data Base.

The results can best be presented in two tables. The first table shows for each test point the codes chosen as B_1 , B_2 , P_1 or P_2 by EFAGHY together with the number of votes each code received. The greatest number of votes is marked with a prime (') and the second greatest is marked with a double prime (''). Columns are marked with an X to signal that the stability condition that actually prevailed was among B_1 , B_2 , P_1 or P_2 , that it was B_1 , that it was P_1 , that B_1 and P_1 agreed irrespective of whether they were correct.

Aeromechanical Compressor/High Aspect Ratio

Point Number	Correct Stability Code	EFAGHY Choice			X if Correct Code is					
		Code	B vote	P vote	Among B ₁ , B ₂ , P ₁ , P ₂			B ₁	P ₁	X if B ₁ = P ₁
1	01	01	9'	5.7"	X			X		
		03	6.5"	6.8'						
2	01	00	9'	7'	X			X	X	X
		01	9'	5.7						
3	02	03	5"	5.9"						
		00	7"	5						
4	02	01	9'	5.7"						
		03	5	5.9'						
5	02	00	7'	5"						
		01	7'	3.9						
6	02	03	5"	5.9'						
		00	7.4"	5.2						
7	02	01	8.4'	5.7"						
		03	5.0	5.9'						
8	02	07	7	5.9"						
		00	7.5"	5.2"						X
9	02	07	9'	7.6'						

Development Fighter Engine, Case 1, Fan Rotor, 1st Stage

Point Number	Correct Stability Code	<u>EFAGHY Choice</u>			<u>X if Correct Code is</u>						
		Code	B vote	P vote	Among B_1, B_2, P_1, P_2				B_1	P_1	X if $B_1 = P_1$
1	00	00	3'	4.9"	X						X
		01	7"	4.0							
		03	9'	7.1'							
2	00	00	5	6.8"	X						X
		01	7"	4.0							
		03	9'	7.1'							
3	01	00	5	6.8"	X						X
		01	7"	4.0							
		03	9'	7.1'							
4	01	00	5	6.8"	X						X
		01	7"	4.0							
		03	9'	7.1'							
14	01	00	5	6.8"	X						X
		01	7	4.0							
		03	9'	7.1'							
23	01	00	5	6.8"	X						X
		01	7"	4.0							
		03	9'	7.1'							
27	01	00	5	6.8"	X						X
		01	7"	4.0							
		03	9'	7.1'							

Development Fighter Engine, Case 2, Fan Rotor, 2nd Stage

Point Number	Correct Stability Code	<u>EFAGHY Choice</u>			<u>X if Correct Code is</u>			X if $B_1 = P_1$
		Code	B vote	P vote	Among B_1, B_2, P_1, P_2	B_1	P_1	
1	01	00	5"	6.8'	X			
		01	5"	2.2				
		03	7'	5.4"				
2	01	00	3	6.8'	X			
		01	5"	2.2				
		03	7'	5.4"				
3	01	00	3	4.9"	X			X
		01	5"	2.2				
		03	7'	5.4'				
4	01	00	3	4.9"	X	X	X	X
		01	7'	4				
		03	7'	5.4'				
		10	5"	0.7				
5	00	00	5.2"	6.8'	X		X	
		05	8.7'	2.9				

Development Fighter Engine, Case 3, Compressor Rotor

Point Number	Correct Stability Code	<u>EFAGHY Choice</u>			<u>X if Correct Code is</u>			X if B ₁ = P ₁
		Code	B vote	P vote	Among B ₁ , B ₂ , P ₁ , P ₂	B ₁	P ₁	
41	00	00	3	4.9"	X			X
		01	5"	2.2				
		03	9'	7.1'				
		22	5"	4.7				
42	01	00	3	4.9"	X			X
		01	5"	2.2				
		03	9'	7.1'				
		22	5"	4.7				
48	00	01	5"	2.2				X
		03	9'	7.1'				
		20	5"	6.6"				
49	01	00	3	4.9"	X			X
		01	5"	2.2				
		03	9'	7.1'				
52	00	01	5"	2.2				X
		03	9'	7.1'				
		20	5"	6.6"				
53	01	00	3	4.9"	X			X
		01	5"	2.2				
		02	5"	4.1				
		03	9'	7.1				
		05	5"	2.8				

High Bypass Turbofan Engine, 20" Smaller

Point Number	Correct Stability Code	<u>EFAGHY Choice</u>			<u>X if Correct Code is</u>				
		Code	B vote	P vote	Among B ₁ , B ₂ , P ₁ , P ₂				X if B ₁ = P ₁
1	07	02	8.6"	6.3"					
		03	7.3	7.0'					
		05	9'	3.0					
2	07	02	7.5'	6.1"					
		03	6.6	6.4'					
		05	7.0"	2.8					
3	00	03	8.5"	7.1'					
		10	8.9'	5.6"					
4	01	03	8.4'	7.1'					X
		10	8.3"	4.7"					
5	01	01	6.0	3.6"	X				X
		02	4.1	3.6"					
		03	8.7'	7.1'					
		10	6.9"	3.0					
6	00	02	5.0	4.5"					X
		03	8.8'	7.1'					
		10	6.9"	3.0					
7	00	01	7.0"	4.0					X
		02	5.0	4.5"					
		03	8.6'	7.1'					
		20	5.0	4.5"					

Rotating Rig Test

Point Number	Correct Stability Code	<u>EFAGHY Choice</u>			<u>X if Correct Code is</u>					X if $B_1 = P_1$
		Code	B vote	P vote	Among B_1, B_2, P_1, P_2				B_1 P_1	
54	02	00	5.5	6.9"						X
		03	8.5'	7.1'						
		10	8.4"	4.2						
79	02	00	5.5	6.9"						X
		03	8.5'	7.1'						
		10	8.4"	4.2						
158	10	00	5.5	6.9"	X					X
		03	8.6'	7.1'						
		10	8.4"	4.2						
56	02	00	5.5"	6.9"						X
		03	8.5'	7.1'						
		10	8.5'	4.2						
89	02	00	5.5"	6.9						X
		03	8.5'	7.1'						
		10	8.5'	4.2						
179	10	00	5.3	6.9"	X					X
		03	8.7'	7.1'						
		10	6.5"	2.3						
154	10	00	5.2	6.8"	X					X
		03	8.8'	7.1'						
		10	6.5"	2.3						
57	03	00	5.3	6.9"	X			X	X	X
		03	8.7'	7.1'						
		10	6.6"	2.3						

Rotating Rig Test

Point Number	Correct Stability Code	<u>EFAGHY Choice</u>			<u>X if Correct Code is</u>				X if $B_1 = P_1$
		Code	B vote	P vote	Among B_1, B_2, P_1, P_2	B_1	P_1		
61	01	00	5.0	6.8'					
		03	7.0'	5.4					
		10	6.8"	2.3					
		20	5.0	6.6"					
169	01	00	5.0	6.8"					
		03	7.0'	5.4					
		10	6.9"	2.3					
		20	5.5	6.9'					
176	10	00	5.0	6.8"	X				
		03	7.0'	5.4					
		10	6.9"	2.3					
		20	5.9	7.1'					
172	10	00	5.0	6.8"	X				
		03	7.0'	5.4					
		10	6.9"	2.3					
		20	5.9	7.1'					
63	01	00	5.0	6.8"					
		03	7'	5.4					
		10	7'	2.3					
		20	6.6"	7.6'					
148	01	00	5.1	6.8"	X				
		01	7'	4.0					
		03	6.9"	4.0					
		10	7'	7.5'					
		20	6.5	7.5'					
151	10	00	5	6.8"	X		X		
		03	7'	5.4					
		10	7'	2.3					
		20	6.7"	7.6'					

High Bypass Turbofan Engine

Point Number	Correct Stability Code	<u>EFAGHY Choice</u>			<u>X if Correct Code is</u>						
		Code	B vote	P vote	Among $B_1, B_2, P_1, P_2, B_1, P_1, B_1 = P_1$						
92	01	01	5"	2.2	X						X
		03	7'	5.4"							
		22	7'	6.1'							
93	01	00	5"	6.8"	X						X
		01	5"	2.2							
		03	9'	7.1'							
		10	5"	0.7							
94	00	00	3	4.9"	X						X
		01	5"	2.2							
		03	7'	5.4'							
		22	5"	4.7							
95	00	00	5"	6.8"	X						X
		01	5"	2.2							
		03	9'	7.1'							
		22	5"	4.7							
96	00	00	5"	6.8"	X						X
		01	5"	2.2							
		03	9'	7.1'							

For the Validation Points B_1 was correct in a literal sense for only 5 points, 10 percent, and P_1 for only 4 points, 8 percent. Furthermore the stability regions chosen as B_1 or P_1 for the other points have been reviewed and found were also not correct in any practical sense, that is the Validation Point is not in or near a transition to the regions chosen as B_1 or P_1 . These percents contrast to 59 percent for EFAGHY applied to the Annular Cascade Data Base.

For 67 percent of the points (34 of 51) the B_1 and P_1 choices agreed, but only 9 percent (3 of 34) were correct. Thus the agreement of B_1 and P_1 is no indicator of a correct answer. For the Data Base B_1 and P_1 agreed for 74 percent of the points and when this occurred their choice was correct for 87 percent of the cases.

For 61 percent of the points (31 of 51) the correct choice was among B_1 , B_2 , P_1 , P_2 . This compares with 84 percent for the Data Base. The conclusion from the above would seem to be that although neither B_1 nor P_1 were proven useful as indicators of the correct stability region, the weaker result stands: that B_1 , B_2 , P_1 , P_2 include the correct result for a fair percentage of the cases. This result would suggest that the hyperplanes have captured some of the reality of the Validation Points.

However even this mildly favorable conclusion seems to be unjustified as the following table shows.

The following table shows for each Validation Point the B_1 , B_2 , P_1 , P_2 choices made by EFAGHY. The Validation Points having the same (correct) stability code are grouped together, without identification, but taken in the order used in the prior table.

128

The principal observation that can be made from the above table is that to the hyperplane/voting system in EFAGHY, the Validation Points all look similar in that the choice 00, 01 and 03 is made for almost every Validation Point. Since for 69% of the Validation Points, the correct choice is actually one of these three, it is to be expected by chance only that the correct choice would occasionally be made.

In fact, the response to certain groups of Validation Points is even more striking. The previous table, for example, shows that for all six Validation Points from the F101 Derivative Fighter Engine, Fan Rotor, 1st Stage the leading B and P scores are all almost identical. This similarity holds also for the lower scores for the other stability regions, not shown in the table.

There are seven such groupings of nearly identical votes for points. In all but one grouping, there is a diversity of actual stability condition. One possible explanation of the discrepancy between the hyperplane/voting system in EFAGHY as applied to the Annular Cascade Data Base and as applied to the Validation Points is that measurement of the nine aeromechanical variables in production engines is not as exact as in the Annular Cascade. To verify whether this was a factor, as well as to check for closeness to transition zones, the Validation Points were rerun with 18 variations, namely each of the 9 aeromechanical variables was perturbed + 10 percent. In most cases the B_1 , B_2 , P_1 , P_2 choices by EFAGHY were the same as before. In hardly any case was the correct answer chosen.

Another possible explanation might be that the Validation Points covered too narrow a range relative to the 891 test points from the Annular Cascade Data Base, so that the Validation Points might all look alike. Alternatively, discrepancies might be explained if the range of the Validation Points exceeded that of the 891 points with many points having variables beyond the range of the 891 points.

The following table shows for the 51 Validation Points the maximum and minimum of each variable. The latter are then expressed relative to the corresponding maximum and minimum of the 891 points from the Annular Cascade Data Base, transformed to a scale from 0 to 1.

	PIN	INLET	SLDTYQ	REY	INC	MLE	VLE875	REDV1	REDV2
Point Number	<u>2</u>	<u>3</u>	<u>6</u>	<u>9</u>	<u>10</u>	<u>11</u>	<u>14</u>	<u>15</u>	<u>16</u>

51 Validation Points

Max	49.3	868	1.63	47.4	17.1	1.334	1447	11.73	3.09
Min	16.3	535	0.81	6.73	3.45	0.424	472	2.61	0.64

51 Validation Points, Relative to Data Base Points

Max	0.54	0.87	0.83	2.42	0.71	0.71	0.64	1.24	0.26
Min	-0.25	0.002	-0.21	0.14	-0.03	0.06	0.05	0.16	0.01

891 Points From Data Base

Max	68.5	920.3	1.762	22.048	16.72	1.398	1658	9.693	9.93
Min	26.8	53.43	.974	4.19	-8.71	0.342	389	1.27	0.51

We can readily see that the range of the variables for the Validation Points is not small relative to the Data Base, but generally occupies a substantial portion within the range of the Data Base variables.

However inlet pressures for 65 percent (33 of 51) of the Validation Points are below the range of the 891 Data Base points; Reynold's numbers for 45 percent are above their Data Base range; solidity for 14% are below their Data Base range.

In fact there are only 4 points whose data all lie within the range of the 891 Data Base points, namely points 61, 169, 176, 172 from the Rotating Rig Test.

The possibility cannot therefore be discounted that the Validation Points do not fit the hyperplanes gotten from the 891 Data Base points because the Validation Points lie partially outside the range of the Data Base points. However the possibility loses some credibility since the hyperplanes do not fit the four Validation Points that lie within the range of the Data Base points any better than those that lie partially outside.

7.0 Resolution of Issues

As a result of the application of the hyperplanes produced by GALACTIC to the Annular Cascade Data Base as well as to validation points not in the data base a number of issues have surfaced. These will now be grouped under eight headings and discussed, together with a plan for the resolution of each issue.

7.1 Correctness of GALACTIC Hyperplanes

In the case of three hyperplane pairs (those that discriminate between stability regions 00 and 21, between regions 00 and 20, and between regions 10 and 23) the first discriminant method declared that the regions were disjoint even in subspaces of only 6 or 7 of the 9 aeromechanical variables. But the Relaxed Discriminant procedure for finding the discriminating planes was unable to do so even by using all 9 variables either for the overlap or gap case - at least within the 2000 allowed iterations. While the two discriminating methods have proven generally consistent, with allowance for numerical differences, these two cases raise the question of a possible bug in the code. The discrepant result in these two cases might however be due to the insufficient iteration or to numerical difference due to the complete diversity of the two methods. In these three cases the latest hyperplane produced was accepted by GALACTIC - which was, perhaps, not the best choice. But when GALACTIC and later EFAGHY tested two of these hyperplanes against the data points upon which they were constructed, we found that the majority of test points in one stability region was misclassified.

Whether there is an error in the coding or simply an inadequacy, it should be possible to improve the code so that not only in theory, but also in practice, a solution will always be found which, at worst, may place many points in an overlap region. Except for borderline cases, no points in the data base should be misclassified. It is believed, however, that this situation has impacted a relatively few hyperplanes and test points, and that the effect is further diluted in the voting process.

7.2 Nonlinear Discriminating Surfaces

While only a few hyperplanes misclassify very many data points from which they were generated, there are 21 which are ineffective in that they place in an ambiguous overlap zone a large number of the test points from which they were constructed. This is attributed to the inability of a plane to separate the two stability regions, a situation in which a curved surface is needed. Fortunately the existing linear code in GALACTIC can be utilized for this purpose, without change. If a nonlinear combination of the nine basic aeromechanical input variables can be suggested on physical grounds, it is an easy matter to enter this into one of the unused words in the 20-word record for each test point and redo certain of the hyperplanes using 10 instead of 9 aeromechanical variables.

The new combination variable might be a plausible flutter parameter, such as researchers have already explored, or it could be a second term in a mathematical expansion of a basic variable. In either case, this is an easily accomplished task which would not require modification of GALACTIC or EFAGHY and would be applied to only the 21 hyperplanes with heavily populated overlap region. It is however not clear how much experimentation with various nonlinear combinations might be required before finding suitable combination variables.

Note that GALACTIC provides little direct help in selecting nonlinear combination variables if the above choices prove inadequate. The testing of such variables, once conceived is however easy. One help that GALACTIC can offer in choosing the combination variables is the identification of clusters that contain, in close proximity, test points from more than one stability region. It can provide rotation of the test space so that the points can be viewed most advantageously. This would identify the linear combination of variables which are playing a key role, but it does not say how to combine these variables nonlinearly.

7.3 Additional Aeromechanical Variables

If we find that after introduction of nonlinear combination variables, certain pairs of stability regions are still not separable, it may be that separation requires use of other than the 9 aeromechanical variables currently employed.

One clear indication of this, even before experimentation with nonlinear combination variables, would be the occurrence of points of one stability region surrounded by points of another. Tools for analysis of clusters and convex hulls currently in GALACTIC would be helpful in this regard.

In such a situation an additional variable could perhaps lift the surrounded point out of its alien environment.

Were this the case, the practical conclusion would be that on production engines additional instrumentation is needed to supply the additional aeromechanical variables, if flutter prediction is required. The implication would be that past attempts to predict flutter empirically have been flawed by inadequate information.

Identification of additional necessary aeromechanical variables could be a major benefit from this effort.

7.4 Data Base Selection

The 891 test points supplied to GALACTIC for construction of the hyperplanes were selected carefully from the thousands of test points in the Annular Cascade Data Base. The points were selected so as to define the transition zones from one stability region to another. If hyperplanes could be found to discriminate between these neighboring points, it seemed plausible that they would also discriminate between points remote from transition zones. Due to computer storage limitations, duration of runs and the cost thereof, it was important to construct the hyperplanes using as few test points as possible.

While the selection logic seems impeccable, it can be easily verified that the selected points are representative namely by applying the hyperplanes to the other points from the Annular Cascade Data Base, those which were not used by GALACTIC to construct the hyperplanes.

This can be readily and cheaply done by EFAGHY since there is virtually no limit on the number of points to which the hyperplanes can be applied.

If we find that the hyperplanes correctly predict the stability region in which the new points are located, then this indicates a basic difference between the Annular Cascade Data Base and the real world Validation Points.

On the other hand, if we find that substantial numbers of the new points from the Annular Cascade Data Base are not correctly assigned, then this indicates that the data base selected for GALACTIC was not adequate.

Should the latter be the case, then the corrective action is clear, namely to include some or all of the incorrectly assigned points among the test points used by GALACTIC and to call upon GALACTIC to recalculate

the hyperplanes which discriminate the stability regions to which these points belong.

Hopefully the new hyperplanes would prove more effective both in classifying the test points of the Annular Cascade Data Base which would still not have been used by GALACTIC, as well as in classifying the Validation Points.

The only difficulty anticipated is that computer storage and running time limitations may be encountered if it is necessary to include many more data points in those stability regions which are already well populated.

7.5 Validity of the Validation Points

There is no question but that the data in the Annular Cascade Data Base are more precise than the real world Validation Points.

To determine if such inaccuracy could account for the incorrect classifications produced by the hyperplanes, the Validation Points were perturbed plus and minus 10 percent, in each of the nine variables individually, in the belief that this would include any possible inaccuracy in the data. The correct stability region was not chosen by EFAGHY in hardly any of the cases. Indeed the choice of stability region did not change very frequently.

While the perturbations were not done in all variables simultaneously, since this would entail 2^9 cases per test point, there is no indication that the correct answer would be chosen more frequently.

An alternate approach might however prove more efficient, namely to inquire how distant the Validation Point is from the nearest test point from its stability region which was used in the construction of the hyperplanes.

To do this would require some small amount of additional coding. But the basic information is already available, namely the distance of the point from each of the hyperplanes intended to bound the stability region to which the point in fact belongs.

7.6 Annular Cascade Data Base Applicability

The ultimate issue is of course the applicability of the Annular Cascade research vehicle to real world engine data.

If the hyperplane system of GALACTIC and EFAGHY works well on the test points from the Annular Cascade Data Base - particularly on test points which GALACTIC did not use - but poorly on the Validation Points, then the issue of artificiality of the Data Base becomes more compelling.

It would be possible to prove such incompatibility if we can show that Validation Points are surrounded by Annular Cascade test points belonging to a different stability region.

All of the foregoing work described in this section can be expected to improve the validity of the hyperplanes. Thus such work can only sharpen the issue of artificiality should the hyperplanes prove ultimately unable to correctly classify the Validation test points.

Should this prove to be the case, the new question arises as to a possible real world correction. Presumably there would be some parameter which had one value for the Annular Cascade and some different values for the Validation Points. There would be of course no clue in the Annular Cascade Data Base as to what this parameter might be as it would have been common to all the test points.

Discovery of such a parameter might arise from engineering reasoning, possibly from among the aeromechanical data recorded for the Annular Cascade but not for production engines. On the other hand, it might be

devised simply as an empirical correction that would shift the stability regions.

A third approach of course would be the incorporation of the current Validation Points into the data base used by GALACTIC, together with identification of new test points with which to validate the hyperplanes. It would be interesting to see which pairs of stability regions would admit hyperplanes which would be violated by neither the Annular Cascade Data Base, nor the Validation Points.

7.7 Implied Operational Map for Validation Points

If it were concluded after the various possible improvements were made, that the Annular Cascade Data Base is not compatible with the Validation Points, it would be possible by use of EFAGHY to construct the operational map for each set of Validation Points, as implied by the Annular Cascade Data Base. If such a map were at all reasonable, it might be that a relation between the actual and the implied map could be discovered. This could lead to a fundamental insight as to which aeromechanical variable requires correction, or an empirical correction.

7.8 Combination Stability Regions

The stability regions as presently defined do not include global regions for example for stall, but only the subregions for the different sub types of stall, such as flexural, torsional etc.

It has been suggested that the hyperplanes might prove more effective if they addressed the question of separating, for example, stall from choke, flutter from stability. Only after this basic question was addressed, would the type of stall be investigated.

Certainly the existing system can be used to discriminate between test points classified into any non-overlapping regions. But this would entail of course an independent set of hyperplanes, since it would not be possible to regard both generic stall and flexural stall in the same analysis as these regions overlap.

It would be worthwhile and inexpensive to try this idea. No new programming would be needed. The only problem would be the large number of test points that might need to be analyzed simultaneously. It is not clear now how limiting that problem might be.

8.0 Conclusions

8.1 Methodology for Analysis of Large Data Bases

The principal conclusion is that a general purpose technology has been developed and implemented in checked out computer code for analysis of large data bases. While this was devised for application to the Annular Cascade Data Base for empirical prediction of flutter, the technology is general purpose, with much broader application.

The most important of these technological developments are

- o Relaxed Discriminant Method for determining two parallel hyperplanes that separate two sets of points
- o Discriminant Feasibility Method to determine whether two sets of points are separable by a hyperplane
- o Stepping Stone Cluster Method to identify clusters using the two metrics: Euclidean and Stepping Stone, without need to quantify in advance what is meant by a small distance.
- o Hyperplane + Voting system to identify regions occupied by various groupings of like points. These regions can then be used to predict the grouping to which a new point would belong.

In addition there were several other developments: the Non-Objective Simplex Method, L1 eigenanalysis, and matrix techniques to determine the shape and size of groupings of points. These methods can be applied to other engineering problems, as well as related maintenance and supply problems, as well as to completely different data bases for example data bases concerned with personnel.

8.2 Hyperplane/Voting System Applied to Annular Cascade Data Base

The above GALACTIC/EFAGHY system has been applied successfully to 891 test points of the Annular Cascade Data Base.

For 83 of these test points whose aeromechanical situation was looked at closely, the system correctly identified the stability condition of 59 percent of the points in a narrow literal sense. But in a practical sense, 79 percent of the points were correctly identified.

When applied to the 891 test points, the system again correctly identified the stability condition of 59 percent of the points in a narrow literal sense. Since it was not possible to look closely at the aeromechanical situation of all of these points, it is not possible to say how many were correct in a broader practical sense. But since the 59 percent correctness in a literal sense agreed for both sample and population, it can be conjectured that the 79 percent correctness in a practical sense might apply not only to the sample but to the population as well.

These already favorable success ratios need to be placed in perspective. First, the test points to which the Hyperplane/Voting system was applied are the same test points used in constructing the system so that high success is to be expected. Second, only 76 percent of the hyperplanes are adequate without nonlinear variables; thus, there are various easily available steps using the existing Hyperplane/Voting system by which the success ratio can be increased further.

8.3 Hyperplane/Voting System Applied to Validation Points

The Hyperplane/Voting System applied to 51 Validation Points taken from seven sets of engine test data, was however, not successful in identifying the stability condition either in a narrow literal sense, nor in a broader practical sense.

This raises questions:

- o whether the Annular Cascade Data Base is representative of real engine data
- o whether the 891 point sample used to construct the hyperplanes was representative of the full Annular Cascade Data Base.

Specific steps are proposed for resolving these questions, and for using the insights thus obtained to improve the predictive ability of the Hyperplane/Voting systems when applied to real engine data.

Appendix A - GALACTIC Listing


```

*5 NINF
C NPMAX,NVMAX,ICLMAX,MUST AGREE HERE,IN BOTTLE,DETERM,DTRMIN,
C PLOTSCAT,INSECT,IVECTR AND NOSMPLX
C APPROXIMATELY AT LINES 50,9070,9670,11070,10170,12100
C 6163 AND 14155
C TO SCREEN DATA FROM FILE, ACTIVATE STATEMENTS AS IN S62,S10U4
PARAMETER (NPMAX=450,NVMAX=14,ICLMAX=14)
PARAMETER (NAXES=NVMAX*ICLMAX,NVNLMAX=NVMAX*(NVMAX+1)/2)
PARAMETER (INMXMX=20,ISETMAX=14)
PARAMETER (NSPAIR=(ISETMAX*(ISETMAX-1))/2,NVP3=NVMAX+3)
PARAMETER (NVERTX2=NPMAX,NSPAIR3=2*NSPAIR+NVNLMAX)
PARAMETER (NPNVMAX=NPMAX*NVMAX+1)
PARAMETER (LWK=(NVMAX+2)*NVERTX2+4*(NVERTX2+4)*2)
PARAMETER (LWKPNV=LWK-NPMAX*NVMAX)
PARAMETER (NFILMAX=14)
COMMON LNPMAX,LNVMAX,LICLMAX,LNAXES,LLWK,WK
DIMENSION WK(LWK)
DIMENSION XINF(NPMAX,NVMAX),SDISTI(NPMAX),SDISTJ(NPMAX)
&.IDISTI(NPMAX),IDISTJ(NPMAX),ICLUSTER(ICLMAX,NPMAX)
&.IFRONTIR(NPMAX),IFRONTIC(NPMAX),IFRONTJIR(NPMAX),
&.IFRONTJIC(NPMAX),AXES(NAXES,NVMAX),CG(ICLMAX,NVMAX),
&.SDISTC(ICLMAX,ICLMAX),STVP(NVMAX),SVAR(NVMAX),IPINCL(NPMAX)
&.NPINCL(ICLMAX),AXLENG(NAXES)
&.IN(NVMAX),VAR(NVMAX,NVMAX),TEMPM(NVNLMAX,NVNLMAX)
&.IBOND(NPMAX),JBOND(NPMAX),CHS(NVNLMAX),NAINCL(ICLMAX)
&.BUFFER(NPMAX),TEMPV(NSPAIR3)
&.NANLINCL(ICLMAX),AXTOT(NVMAX,NVMAX),AXLTOT(NVMAX)
&.Q(NVMAX),ISETCOM(ISETMAX),ISETCOT(ISETMAX),IJJ(NVNLMAX)
&.IJJ(NVNLMAX),IPINSET(NPMAX),NPINSET(ISETMAX),IBUFFER(NPMAX)
&.ISETID(ISETMAX),ITEMPV(ICLMAX),ISETCL(ISETMAX,ICLMAX)
&.CGTOT(NVMAX),VARMOM(NVMAX,6),IBUFFER1(NPMAX),P(NVP3,NSPAIR)
&.ITEMP(NVNLMAX,NVNLMAX),ITEMPMV(1),TEMPMV(1),NVINSET(ISETMAX)
&.ISETOUT(ISETMAX),ISETOUTT(ISETMAX),IPTOUT(20),IPTOUTT(20)
&.NPINSETT(ISETMAX),NPIN(ICLMAX),VARMMU(INMXMX,6)
&.HCONL(NSPAIR),HCONR(NSPAIR)
EQUIVALENCE (TEMPM,TEMPMV),(ITEMPM,ITEMPMV)
EQUIVALENCE (BUFFER,IBUFFER1),(JBOND,IBUFFER1),(TEMPM,ITEMPM)
EQUIVALENCE (XINF(1,1),SDISTI),(XINF(1,2),SDISTJ)
&.XINF(1,3),IDISTI),(XINF(1,4),IDISTJ)
&.XINF(1,5),IFRONTIR),(XINF(1,6),IFRONTIC)
&.XINF(1,7),IFRONTJIR),(XINF(1,8),IFRONTJIC)
&.XINF(1,1),ICLUSTER(1,1))
EQUIVALENCE (WK(NPNVMAX),TEMPM(1,1))
EQUIVALENCE (WK(LWKPNV),XINF(1,1))
CHARACTER*8 DATE1
CHARACTER*64 FILIN(NFILMAX),FILINT,FILINR,FILECR,FILEHI,FILEHO
CHARACTER*9 FILENAME
CHARACTER*1CHS,CHP,CHM,CHU,CHV,CHW,CHX,CHY,CHZ,CHN,CHO
&.CHB,CHC,CHD,CHL,CHM,CHN,CHO
&.CHGT,CHHQ,CHLT,CHNO
CHARACTER*3CHYN,CHNOT,CHBL
CHARACTER*3CH*AR(13),CHTYP,CHSCL
CHARACTER*7CHSETCL,CHSET,CHCLS,CHMODE,CHBAT,CHTSH
CHARACTER*1BREM,F
LOGICAL EXIS
INTEGER H
DOUBLE PRECISION IOUTPUT,IOUTPS,IOUTPC,IOUTPSC,IDIGIT
DATA (CH*AR(1),1=1,13)/MIN,'MAX','AVE','SIG','MID','RAN',
&.SPE,'MNU','MAX','ASU','SGU','MDU','RNU'/'
ISTOP=0

```

```

CALL MEMSIZ(J)
C IF MEMSIZ IS THE VAX DUMMY MEMSIZ ROUTINE THAT SIMPLY SETS J=10,
C DO NOT PRINT OUT THE NUMBER OF WORDS ALLOCATED
IF (J.NE.10) WRITE(06,*)1024*J,' WORDS ALLOCATED'
IF(NVNLMAX.GE.2*NVMAX.AND.NVNLMAX.GE.20.AND.NVNLMAX*2.GE.32.*
&NVMAX*ISETMAX.AND.NVNLMAX.GE.NSPAIR*4) GO TO 10
WRITE(06,*)' PARAMETER NVNLMAX=' NVNLMAX,' BUT MUST BE GE 2*
&NVMAX=' 2*NVMAX,' AND GE 20 AND GE SORT(32*NVMAX*ISETMAX))=,
&SQRT(32.*FLOAT(NVNLMAX)*FLOAT(ISETMAX)),
&' AND GE ISETMAX*(ISETMAX-1)+NVMAX =' ,NSPAIR*2+NVMAX
ISTOP=1
10 IF(NAXES.GE.NVMAX) GO TO 20
WRITE(06,*)' PARAMETER NAXES=',NAXES,' BUT MUST BE GE NVMAX='
&NVMAX
ISTOP=1
20 IF(ICLMAX.GE.10.AND.ICLMAX.LE.NVMAX) GO TO 30
WRITE(06,*)' PARAMETER ICLMAX =' ,ICLMAX,' BUT MUST BE AT
& LEAST 10 AND LE NVMAX=' ,NVMAX
ISTOP=1
30 IF(NVMAX.GE.9.AND.NVMAX.GE.ISETMAX) GO TO 40
WRITE(06,*)' PARAMETER NVMAX=' ,NVMAX,' BUT MUST BE AT LEAST 9
& AND GE', ISETMAX
ISTOP=1
40 LNPMAX=NPMAX
LNVMAX=NVMAX
LICLMAX=ICLMAX
LNAXES=NAXES
LWK=LWK
LWKMAX=2*NPMAX*NVMAX*MAX0(8*NVMAX*ISETMAX,2*NVERTX2*NVMAX+
&(NVMAX*4)*(NVMAX+1))-1)
IF(LWK.GE.LWKMAX) GO TO 50
WRITE(06,*)' LWK DIMENSION',LWK,' LT',LWKMAX,' WORDS NEEDED TO
& OVERLAY WITH ANU,TEMPMV/ITEMPV/TEMPM/ITEMPM,XINF'
ISTOP=1
50 IF(ICLMAX.GE.ISETMAX) GO TO 55
WRITE(06,*)' PARAMETER ICLMAX=' ,ICLMAX,' BUT MUST BE AT LEAST
& PARAMETER ISETMAX=' , ISETMAX
ISTOP=1
55 IF(ISTOP.NE.U) STOP
WRITE(06,*)' 219 LWK,LWKMAX,LWK,LWKMAX
IBITS=2**30-1
CHP='+'
CHM='-'
CHU='U'
CHV='V'
CHW='W'
CHY='Y'
CHN='N'
CHB='B'
CHE='E'
CHBLT=' '
CHO='O'
CHGT='>'
CHEQ='='
CHLT='<'
CHNO='/'
CHNOT='NOT'
CHBL=' '
CHSET=' SET
CHCLS=' CLUSTR

```

```

CHBAT= BATCH
CHTSH= TSHARE
FILENAME=
FILECR=
ICASE=0
VBIGNO=1.E30
60 FORMAT(15I4)
70 FORMAT(10F7.3)
75 FORMAT(15,13,16F7.3)
80 FORMAT(10I7)
90 FORMAT(1X,4E16.8)
100 FORMAT(A2,'(',12,')'=E11.4,A2,E10.3,'*',A1,'(',12,')',
&A2,E10.3,'*',A1,'(',12,')',A2,E10.3,'*',A1,'(',12,')',
110 FORMAT(A2,E10.3,'*',A1,'(',12,')',A2,E10.3,'*',A1,'(',12,')',
&A2,E10.3,'*',A1,'(',12,')',A2,E10.3,'*',A1,'(',12,')',
C=====
IMODE=MODE(1)
DATA (VARMU(1,JJ),JJ=1,6)/26.8,68.5,51.649194,13.450007
&.47,65.41,7/
DATA (VARMU(2,JJ),JJ=1,6)/534.3,920.3,780.78507,132.25561
&.727,3.386,/
DATA (VARMU(3,JJ),JJ=1,6)/10.38,23.23,17.7359,2.4587139
&.16,805.12,85/
DATA (VARMU(4,JJ),JJ=1,6)/-.34,3.63,2.0190405,.63665905
&.1,645.3,97/
DATA (VARMU(5,JJ),JJ=1,6)/.974,1.762,1.2189113,.21632369
&.1,368.,788/
DATA (VARMU(6,JJ),JJ=1,6)/393.,68.949,58.23621,4.8171529
&.54,1265,29.645/
DATA (VARMU(7,JJ),JJ=1,6)/.344,1.457,.82604469,.18210604
&.9005,1.113/
DATA (VARMU(8,JJ),JJ=1,6)/4.19,22.048,9.9088658,3.6629626
&.13,119,17.858/
DATA (VARMU(9,JJ),JJ=1,6)/-8.71,16.72,6.0870879,4.1214311
&.4,005,25.43/
DATA (VARMU(10,JJ),JJ=1,6)/.342,1.398,.79005449,.17107861
&.87,1.056/
DATA (VARMU(11,JJ),JJ=1,6)/.91,2.441,1.3140142,.24496324
&.1,6755,1.531/
DATA (VARMU(12,JJ),JJ=1,6)/-.02,.24,.10449291,.038614039
&.11,.26/
DATA (VARMU(13,JJ),JJ=1,6)/389.,1658.,1007.4842,207.07176
&.1023,5.1269,/
DATA (VARMU(14,JJ),JJ=1,6)/1.27,9.693,4.7684722,1.3461923
&.5,4815,8.423/
DATA (VARMU(15,JJ),JJ=1,6)/.51,9.9299999,1.7055396,.51473065
&.5,22,9.42/
DATA (VARMU(16,JJ),JJ=1,6)/0.5,...39638713,1.1982756,2.5,5./
C=====
120 WRITE(06,140)
130 CALL DATIM(1,TIME1)
140 FORMAT(72(1H=))
CALL PTIME(TIMEIN)
WRITE(06,*)
CALL MESSAGE (1)
CHMODE=CHTSH
IF(IMODE.EQ.0) CHMODE=CHBAT
WRITE(06,150)DATE1,TIME1,CHMODE
150 FORMAT(' RUN ON ',A8,' AT ',F6.2,' HOURS ON ',A7)
WRITE(06,*)

```

```

C=====
CALL MESSAGE (2)
DO 155 K=1,NFILMAX
155 FILIN(K)=FILECK
READ(05,*)NFILE,(FILIN(1),I=1,MIND(NFILE,NFILMAX-1)),INMAX,
& CHCF
IF((MODE.EQ.0).WRITE(06,*)NFILE,',(FILIN(1),I=1,MIND(NFILE,
&NFILMAX-1)),INMAX,','CHCF
IF(NFILE.GT.NFILMAX-1) WRITE(06,*)' CAN ACCEPT ONLY FIRST',
&NFILMAX-1
&,' OF 'NFILE' FILES OFFERED'
REMOVF='REMOVE CLEARFILES'
IF(CHCF.EQ.CHY.AND.IMODE.EQ.1)CALL CALLSS(REMOVF)
C SET UP DATA FILES, DESIGNATION 11 FOR EDIST, 12&22 FOR SDIST1
C IDIST, 14& 13&23 FOR24 FOR IFRONT BY ROW, 15&25 FOR IFRONT BY COL
C EACH HAS NP RECORDS, NP WORDS LONG.
C DESIGNATION 10 FOR XINF, HAVING NP RECORDS, NV WORDS LONG.
ISTAT=0
ITEMPO=NPMAX*NVMAX
ITEMP=NPMAX*NPMAX
OPEN(10,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
IF (ISTAT.NE.0) THEN
WRITE(06,*)' ERROR OPENING XINF FILE. ISTAT=',ISTAT
STOP 1
END IF
OPEN(11,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
IF (ISTAT.NE.0) THEN
WRITE(06,*)' ERROR OPENING EDIST FILE. ISTAT=',ISTAT
STOP 1
END IF
OPEN(12,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
OPEN(22,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
IF (ISTAT.NE.0 .OR. ISTAT1.NE.0) THEN
WRITE(06,*)' ERROR OPENING SDIST FILES. ISTAT=',ISTAT,
& ISTAT1=',ISTAT1
STOP 2
END IF
OPEN(13,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
OPEN(23,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
IF (ISTAT.NE.0 .OR. ISTAT1.NE.0) THEN
WRITE(06,*)' ERROR OPENING IDIST FILES. ISTAT=',ISTAT,
& ISTAT1=',ISTAT1
STOP 3
END IF
OPEN(14,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
OPEN(24,STATUS='SCRATCH',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='UNFORMATTED')
IF (ISTAT.NE.0 .OR. ISTAT1.NE.0) THEN
WRITE(06,*)' ERROR OPENING IFRONT FILES. ISTAT=',ISTAT,
& ISTAT1=',ISTAT1
STOP 4
END IF
OPEN(15,STATUS='SCRATCH',IOSTAT=ISTAT,

```

```

      RACCESS='SEQUENTIAL',FORM='UNFORMATTED')
      OPEN(25,STATUS='SCRATCH',IOSTAT=ISTAT1,
      RACCESS='SEQUENTIAL',FORM='UNFORMATTED')
      IF (ISTAT.NE.0 .OR. ISTAT1.NE.0) THEN
        WRITE(06,*) ERROR CREATING IFRONTC FILE. ISTAT=,ISTAT,
        & STAT1,ISTAT1
      STOP 5
    END IF
  C=====
  210 DO 220 I=1,ISETMAX
    NPINSET(I)=0
    ISETID(I)=IBITS
    ISETOUT(I)=0
  220 CONTINUE
  230 IPINSET(I)=0
    IOUTPUT=0
    IOUTPS=0
    IOUTPC=0
    NSETOUT=0
    NPTOUT=0
    CALL MEMSIZ(J)
  C IF MEMSIZ IS THE VAX DUMMY MEMSIZ ROUTINE THAT SIMPLY SETS J=10,
  C DO NOT PRINT OUT THE NUMBER OF WORDS ALLOCATED
    IF (J.NE.10) WRITE(06,*)1024*J,' WORDS ALLOCATED'
    WRITE(06,250)MINO(INMAX,NVMAX)
  250 FORMAT(' ENTER N (LE,14,'). THEN NAME N VARIABLES')
    READ(05,*)NV,(IN(J),J=1,MINO(NVMAX,NV))
    IF(IMODE.EQ.0)WRITE(06,*)NV,(IN(J),J=1,MINO(NVMAX,NV))
    IF(NV.GT.NVMAX)WRITE(06,*) ERROR, ACCEPTING ONLY FIRST,NVMAX
    & VARIABLES
    CALL MESSAGE (3)
    READ(05,*)ISETVAR
    IF(IMODE.EQ.0)WRITE(06,*)ISETVAR
    IF(NV.GT.0) GO TO 270
  255 IF(IMODE.EQ.0) GO TO 259
  257 WRITE(06,260)
    GO TO 120
  259 STOP 10
  260 FORMAT(' IMPOSSIBLE INPUT INTERPRETED AS CUE TO RESTART RUN')
  270 NV=MINO(NV,NVMAX)
    DO 275 J=1,NV
      IF(IN(J).LE.0) GO TO 255
  275 CONTINUE
    MAXJ=IN(1)
    DO 280 J=1,NV
      MAXJ=MAX0(MAXJ,IN(J))
  280 CONTINUE
    IF(MAXJ.GT.INMAX)WRITE(06,*) ERROR.INPUT FILE HAS ONLY,INMAX
    & VARIABLES
  290 WRITE(06,*) ENTER N (LE,ISETMAX,') THEN ID OF N SETS TO BE
    & EXCLUDED
    READ(05,*)NSETOUT,(ISETOUT(I),I=1,MINO(NSETOUT,ISETMAX))
    IF(IMODE.EQ.0)WRITE(06,*)NSETOUT,(ISETOUT(I),I=1,MINO(NSETOUT
    & ISETMAX))
    IF(NSETOUT.GT.ISETMAX)WRITE(06,295)ISETMAX
  295 FORMAT(' DATA IGNORED BEYOND DIMENSION LIMIT OF ,14)
    IF(NSETOUT.LT.0.AND.IMODE.EQ.1) GO TO 257
    IF(NSETOUT.LT.0.AND.IMODE.EQ.0) STOP 11
    CALL MESSAGE (4)

```



```

      READ(05,*)NPTOUT,(IPTOUT(I),I=1,MIND(NPTOUT,20))
      IF(IMODE.EQ.0)WRITE(06,*)NPTOUT,(IPTOUT(I),I=1,MIND(NPTOUT,
      & 20))
      IF(NPTOUT.GT.20)WRITE(06,295)20
      IF(NPTOUT.LT.0.AND.IMODE.EQ.1) GO TO 257
      IF(NPTOUT.LT.0.AND.IMODE.EQ.0) STOP 14
      CALL MESSAGE(5)
      READ(05,*)NSETCOM,(ISETCOM(I),I=1,2*MIND(NSETCOM,ISETMAX))
      IF(IMODE.EQ.0)
      & WRITE(06,*)NSETCOM,(ISETCOM(I),I=1,2*MIND(NSETCOM,ISETMAX))
      NP=0
      NPTEMP=0
      NPT=0
      ITEMP0=0
      ISETNAM=0
      C NPTEMP=# PTS READ, NPT=# PTS NOT EXCLUDED BY USER, NP=# ANALYZED
      & NUMBERS FOR INCLUDED POINTS
      NFILE=1
      CALL ATFILE(FILIN,NFILE)
      GO TO 320
300 DO 310 J=1,NV
      TEMP=BUFFER(IN(J))
      VARMOM(J,1)=TEMP
      VARMOM(J,2)=TEMP
      VARMOM(J,3)=TEMP
      VARMOM(J,4)=TEMP**2
310
      C
320 READ(01,*,END=360)(BUFFER(L),L=1,INMAX)
      C
      NPTEMP=NPTEMP+1
      ITEMP=BUFFER(ISETVAR)
      IF(NSETOUT.EQ.0.AND.NPTOUT.EQ.0.AND.NSETCOM.EQ.0) GO TO 335
      IF(NSETOUT.LE.0) GO TO 325
      DO 324 K=1,NSETOUT
      IF(ITEMP.EQ.ISETOUT(K)) GO TO 320
324 CONTINUE
      DO 327 K=1,NPTOUT
      IF(NPTEMP.EQ.IPTOUT(K)) GO TO 320
327 CONTINUE
      DO 329 IF(NSETCOM.LE.0) GO TO 329
      DO 327 K=1,NPTOUT
      IF(NPTEMP.EQ.IPTOUT(K)) GO TO 320
329 CONTINUE
      DO 331 K=1,NSETCOM
      IF(ITEMP.EQ.ISETCOM(2*K)) ITEMP=ISETCOM(2*K-1)
331 CONTINUE
      ITEMP0=ITEMP0+1
      ITEMPMV(ITEMP0)=NPTEMP
      IF(ITEMP0.LT.15) GO TO 335
      WRITE(06,60)(ITEMPMV(J),J=1,15)
      ITEMP0=0
335 NPT=NPT+1
      IF(NPT.GT.NPMAX) GO TO 336
      NP=NP+1
      DO 330 J=1,NV
      XINF(NP,J)=BUFFER(IN(J))
      IF(ISETVAR.GT.0) IPINSET(NP)=ITEMP
      IF(ISETVAR.LE.0) IPINSET(NP)=1
336 DO 340 J=1,NV
      TEMP=BUFFER(IN(J))
      VARMOM(J,1)=AMIN1(VARMOM(J,1),TEMP)

```

```

VARMOM(J,2)=AMAX1(VARMOM(J,2),TEMP)
VARMOM(J,3)=VARMOM(J,3)+TEMP
VARMOM(J,4)=VARMOM(J,4)+TEMP**2
340 CONTINUE
IF(ISETNAM.EQ.0) GO TO 346
DO 345 I=1,ISETNAM
IF(BUFFER(ISETVAR).EQ.ISETID(I)) GO TO 349
345 CONTINUE
346 ISETNAM=ISETNAM+1
I=ISETNAM
ISETID(I)=BUFFER(ISETVAR)
349 NPINSET(I)=NPINSET(I)+1
IF(NP.EQ.1) GO TO 300
IF(NPT.EQ.NPMAX+1) WRITE(06,350)NPMAX
350 FORMAT(' WARNING. CAN ANALYZE ONLY FIRST',16,' POINTS. ')
GO TO 320
C
360 NFILE=NFILE+1
CALL ATFILE(FILIN,NFILE)
IF(NFILE.NE.0) GO TO 320
IF(ITEMPO.NE.0) WRITE(06,00)(ITEMPMV(J),J=1,ITEMPO)
IF(NP.EQ.NPT) GO TO 369
WRITE(06,*)
CALL MESSAGE(6)
WRITE(06,3493)'SET ID',(ISETID(I),I=1,ISETNAM)
WRITE(06,349J)'POINTS PER SET',(NPINSET(I),I=1,ISETNAM)
WRITE(06,*)
369 DO 365 I=1,ISETNAM
NPINSET(I)=0
365 ISETID(I)=0
ISETNAM=1
DO 380 I=2,NP
DO 370 J=1,I
IF(IPINSET(I).EQ.IPINSET(J)) GO TO 380
370 CONTINUE
ISETNAM=ISETNAM+1
380 CONTINUE
IF(ISETNAM.LE.ISETMAX) GO TO 400
WRITE(06,390)ISETNAM,ISETMAX
390 FORMAT(' NUMBER OF SUBSETS IS GE',I4,'. EXCEEDS DIMENSION',I4)
GO TO 9000
400 ISETNAM=I
ISETID(I)=IPINSET(I)
IPINSET(I)=1
NPINSET(I)=1
DO 430 I=2,NP
DO 410 J=1,ISETNAM
IF(ISETID(J).EQ.IPINSET(I)) GO TO 420
410 CONTINUE
IF(ISETNAM.GE.ISETMAX) GO TO 440
ISETNAM=ISETNAM+1
J=ISETNAM
ISETID(J)=IPINSET(I)
420 IPINSET(J)=J
NPINSET(J)=NPINSET(J)+1
430 CONTINUE
GO TO 450
440 WRITE(06,*)' NUMBER OF SUBSETS',ISETNAM+1,'EXCEEDS DIMENSION'
& ISETMAX
GO TO 9000

```

```

C 45C XNP=NPT
DO 460 J=1,NV
  VARMOM(J,3)=VARMOM(J,3)/XNP
  VARMOM(J,4)=VARMOM(J,4)/XNP-VARMOM(J,3)**2
  IF(VARMOM(J,4).LT.0.)WRITE(06,*)' CALCULATED VARIANCE IS
  & NEGATIVE: ',VARMOM(J,4), ' ABS VALUE USED IN SIGMA'
  VARMOM(J,4)=SQRT(ABS(VARMOM(J,4)))
  VARMOM(J,5)=(VARMOM(J,1)+VARMOM(J,2))/2.
  VARMOM(J,6)=VARMOM(J,2)-VARMOM(J,1)
460 CONTINUE
470 CALL MESSAGE (7)
  READ(05,*)CHTYP,CHSCL
  IF(IMODE.EQ.0)WRITE(06,*)CHTYP,CHSCL
  DO 480 I=1,13
    IF(CHTYP.EQ.CHVAR(I))I1=I
    IF(CHSCL.EQ.CHVAR(I))I2=I
480 CONTINUE
  IF(I1.NE.0.AND.I2.NE.0) GO TO 490
  WRITE(06,*)' MUST USE ONLY LISTED OPTIONS. USED: ',CHTYP,' ',
  & CHSCL
  GO TO 470
490 IF(I1.NE.7.AND.I2.NE.7) GO TO 520
495 WRITE(06,*)' SUMMARY OF EACH VARIABLE FOR THE ',NPT,' INPUT
  & DATA POINTS'
  CALL MESSAGE (6)
  DO 510 J=1,NV
    WRITE(06,500)J,VARMOM(J,1),
    & VARMOM(J,5),VARMOM(J,2),VARMOM(J,6),VARMOM(J,3),VARMOM(J,4)
510 CONTINUE
  WRITE(06,*)
  IF(IDIGIT(IOUTPUT,IOMAX).NE.0) GO TO 640
500 FORMAT(14,E13.4,5E11.4)
520 IF(I1.EQ.7) GO TO 540
  DO 530 J=1,NV
    IF(I1.LT.7)STYP(J)=VARMOM(J,I1)
    IF(I1.GT.7) STYP(J)=VARMOMU(IN(J)-1,I1-7)
530 CONTINUE
  GO TO 550
540 WRITE(06,*)' ENTER TYPICAL VALUES FOR THE ',NV,' VARIABLES'
  READ(05,*)(STYP(I),I=1,NV)
  IF(IMODE.EQ.0)WRITE(06,*)(STYP(I),I=1,NV)
550 IF(I2.EQ.7) GO TO 570
  DO 560 J=1,NV
    IF(I2.LT.7) SVAR(J)=VARMOM(J,I2)
    IF(I2.GT.7) SVAR(J)=VARMOMU(IN(J)-1,I2-7)
560 CONTINUE
  GO TO 580
570 WRITE(06,*)' ENTER SCALE VALUES FOR THE ',NV,' VARIABLES'
  READ(05,*)(SVAR(I),I=1,NV)
  IF(IMODE.EQ.0,*)WRITE(06,*)(SVAR(I),I=1,NV)
  C
580 WRITE(06,*)
  CALL MESSAGE (9)
  READ(05,*)NBOND
  IF(IMODE.EQ.0)WRITE(06,*)NBOND
  DO 590 I=1,NPMAX
    IBOND(I)=0
590 IBOND(I)=0
  IF(NBOND.EQ.0) GO TO 630

```

```

IF(NBOND.LT.0) GO TO 255
NBOND=MIND(NP,NBOND)
WRITE(06,*) ' ENTER THE ,NBOND, , PAIRS OF POINT NUMBERS'
K=0
600 READ(05,*)I,J
IF(IMODE.EQ.0)WRITE(06,*)I,J
IF(I.GT.0.AND.J.GT.0) GO TO 605
IF(IMODE.EQ.1) GO TO 120
STOP 15
605 IF(I.LE.NP.AND.J.LE.NP) GO TO 610
WRITE(06,*) ' PAIR IGNORED SINCE I OR J EXCEEDS MAX PT NUMBER',NP
GO TO 600
610 I1=MIND(I,J)
J=MAX0(I,J)
I=I1
IF(IBOND(I).EQ.0.AND.IBOND(J).EQ.0) GO TO 620
WRITE(06,*) ' PAIR IGNORED SINCE OVERLAP WITH A PRIOR PAIR'
GO TO 600
620 IBOND(I)=J
K=K+1
IF(K.LT.NBOND) GO TO 600
630 IOMAX=6
IOMAXS=12
IOMAXC=12
WRITE(06,*) ' ENTER 0 OR 1 TO NOTWRITE,OR WRITE FOLLOWING
& OUTPUT OPTIONS:'
WRITE(06,*) ' FOR ALLDATA:SMN,NTR,NPT,BAX,BTR,BPT'
READ(05,*)IOUTPUT
IF(IMODE.EQ.0)WRITE(06,*)IOUTPUT
IF(IOUTPUT.LT.0) GO TO 255
WRITE(06,*) ' FOR SETDATA:IPT,MEM,DSJ,HPL,BAX,PRJ,BTR,CGR,EST,
&HPP,HPO'
READ(05,*)IOUTPS
IF(IMODE.EQ.0)WRITE(06,*)IOUTPS
IF(IOUTPS.LT.0) GO TO 255
WRITE(06,*) ' FOR CLUSTRS:EDS,SDS,STP,FPT,MEM,CCD,BAX,PRJ,BTR,CGR,
&COR,SUR'
READ(05,*)IOUTPC
IF(IMODE.EQ.0)WRITE(06,*)IOUTPC
IF(IOUTPC.LT.0) GO TO 255
IF(IDIGIT(IOUTPUT,IOMAX).EQ.0) GO TO 640
IF(NP.LT.NPT) WRITE(06,*) ' FOR THE POINTS TO BE ANALYZED:'
WRITE(06,*)
WRITE(06,*) ' SET NUMBER SET ID
& NO.OF POINTS
DO 635 K=1,ISETNAM
WRITE(06,*)K,ISETID(K),NPINSET(K)
635 CONTINUE
WRITE(06,*)
GO TO 495
C=====
640 IF(IDIGIT(IOUTPUT,IOMAX-1).EQ.0) GO TO 690
WRITE(06,*)
WRITE(06,*) ' DATA TRANSFORM TO NORMALIZED AXES (V) FROM ORIGINAL
& AXES (U)
DO 650 I=1,N*
IF(SVAR(I).GE.0.)CHS(I)=CHP
IF(SVAR(I).LT.0.)CHS(I)=CHM
650 CONTINUE
DO 660 I=1,N,

```

```

WRITE(06,670)CHV,1,-STYP(1)/SVAR(1),CHS(1),ABS(1)/SVAR(1),
&CHU,1
660 CONTINUE
670 FORMAT(A4,'( ',I2,' ) = ',E11.4,A2,E11.4,'*',A2,'( ',I2,' )')
WRITE(06,*)
WRITE(06,*) DATA TRANSFORM TO ORIGINAL AXES (U) FROM NORMALIZED
& AXES (V)
DO 680 I=1,NV
WRITE(06,670)CHU,1,STYP(1),CHS(1),ABS(SVAR(1)),CHV,1
680 CONTINUE
690 DO 700 J=1,NV
700 LGTOT(J)=(VARMOM(J,3)-STYP(J))/SVAR(J)
IF(IGIT(IGOUTP,IOMAX-2).EQ.0) GO TO 710
WRITE(06,*)
WRITE(06,*) NORMALIZED DATA POINTS, WITH POINT, SET NUMBER AT
& BEGINNING OF ROW
710 DO 730 I=1,NP
DO 720 J=1,NV
XINF(I,J)=(XINF(I,J)-STYP(J))/SVAR(J)
720 CONTINUE
WRITE(10,ERR=740)(XINF(I,J),J=1,NV)
IF(IGIT(IGOUTP,IOMAX-2).EQ.0) GO TO 730
WRITE(06,75)I,IPINSET(I),(XINF(I,J),J=1,NV)
730 CONTINUE
NVR=NV
REWIND 10
GO TO 750
740 WRITE(06,*) ERROR IN WRITING XINF FILE
STOP 6
C COMPUTE EDIST MATRIX OF EUCLIDEAN DISTANCES BETWEEN POINT., NP*NP
750 IF(IGIT(IGOUTP,IOMAX).NE.0) WRITE(06,*)
IF(IGIT(IGOUTP,IOMAX).NE.0)WRITE(06,*) MATRIX OF EUCLIDEAN
& DISTANCES BETWEEN NORMALIZED POINTS
SDISTMX=0.
ITEMP=ISETNAM**2
DO 755 I=1,ITEMP
755 TEMPMV(I)=-1.
DO 760 I=1,NP
II=IPINSET(I)
DO 770 J=1,NP
JJ=IPINSET(J)
IJ=II*(JJ-1)+ISETNAM
JI=JJ*(II-1)+ISETNAM
BUFFER(J)=0.
DO 760 K=1,NV
760 BUFFER(J)=BUFFER(J)+(XINF(I,K)-XINF(J,K))**2
BUFFER(J)=SQRT(BUFFER(J))
IBUFFER(J)=1
SDISTMX=AMAX1(SDISTMX,BUFFER(J))
IF(TEMPMV(IJ).EQ.-1) GO TO 765
IF(TEMPMV(IJ).LT.BUFFER(J).AND.II.EQ.JJ) GO TO 765
IF(TEMPMV(IJ).GT.BUFFER(J).AND.II.NE.JJ) GO TO 765
GO TO 770
765 TEMPMV(IJ)=BUFFER(J)
TEMPMV(JI)=BUFFER(J)
ITEMPMV(ITEMP+IJ)=I
ITEMPMV(ITEMP+JI)=J
770 CONTINUE
BUFFER(I)=0.
IBUFFER(I)=0

```

```

IF(IDIGIT(IOUTPC,IOMAXC).NE.0.AND.NP.GT.10)WRITE(06,*)' ROW',I
IF(IDIGIT(IOUTPC,IOMAXC).NE.0) WRITE(06,70)(BUFFER(J),J=1,NP)
WRITE(11,ERR=780)(BUFFER(J),J=1,NP)
WRITE(12,ERR=780)(BUFFER(J),J=1,NP)
WRITE(13,ERR=780)(IBUFFER(J),J=1,NP)
GO TO 790
780 WRITE(06,*)' ERROR IN WRITING EXIST,SDIST OR IDIST FILE'
STOP 6
790 CONTINUE
REWIND 11
REWIND 12
REWIND 13

IF(IDIGIT(IOUTPS,IOMAXS-9).EQ.0) GO TO 809
WRITE(06,*)' EUCLIDEAN DISTANCES BETWEEN SETS (ON DIAG=WITHIN,
&OFF DIAG=BETWEEN)'
DO 802 II=1,ISETNAM
WRITE(06,70)(TEMPMV(II+(JJ-1)*ISETNAM),JJ=1,ISETNAM)
802 CONTINUE
WRITE(06,*)' END POINTS FOR ABOVE DISTANCES (ONLY ONE FOR WITHIN
& SET)'
DO 804 II=1,ISETNAM
DO 806 JJ=1,ISETNAM
WRITE(06,80)(ITEMPMV(ITEMP+II+(JJ-1)*ISETNAM),JJ=1,ISETNAM)
804 CONTINUE
WRITE(06,*)' POINT SET POINT SET DISTANCE'
DO 808 I=1,NP-1
II=IPINSET(I)
JJ=(II-1)*ISETNAM
READ(11)(BUFFER(K),K=1,NP)
DO 808 J=I+1,NP
JJ=JPINSET(J)
IF(BUFFER(K).GE. TEMPMV(JJ+II+1,OR.II.EQ.JJ)) GO TO 806
WRITE(06,807)I,II,J,JJ,BUFFER(J)
806 CONTINUE
807 FORMAT(I6,I4,I6,I4,F7.3)
808 CONTINUE
REWIND 11
809 DO 812 I=1,8*ISETNAM*NVMAX
812 ITEMPMV(I)=0
CALL VERTEX (XINF,NP,NV,NPMAA,N-MAX,IPINSET,ISETNAM,TEMPMV,
&ITEMPMV)
CHSETCL=CHSET
NSETCL=ISETNAM
IOUTPSC=IOUTPS
IOUTSLC=IOMAXC-IOMAXS-2
DO 900 I=1,ISETNAM
900 NPIN(I)=NPINSET(I)
GO TO 705
C=====
C
C=====
970 DO 972 L=1,NV
DO 971 K=1,ISETNAM
IF(AXLENG(L*(K-1)*NV).GE.AXLENG((1+(K-1)*NV)*.05)) GO TO 972
971 CONTINUE
GO TO 974
972 CONTINUE
975 CALL VERTEX(XINF,NP,L,NPMAA,NVMAX,NPINSET,ISETNAM,
&ITEMPMV(1+2*NV),ITEMPMV(1+2*NV))
DO 980 I=1,NP

```

```

      READ(10,END=985,ERR=985)(XINF(I,J),J=1,NV)
980 CONTINUE
      REWIND 10
      GO TO 930
985 WRITE(06,*) ' STOP. IO ERROR IN READING XINF FILE '
      STOP 7

C
990 AXLTOT(1)=0.
      CGTOT(1)=0.
      AXTOT(1,1)=0.
      LINKS BEGINS =====
      DO 810 J=1,NV
      DO 800 I=1,NP
800 XINF(I,J)=XINF(I,J)-CGTOT(J)
810 AXLTOT(J)=0.
      CALL BOTTLE(XL,F(1,1),AXES(1,1),AXLTOT(1),NV,NP)
      DO 820 I=1,NV
      IF (AXLTOT(1).NE.0.)NAXTOT=I
820 CONTINUE
      DO 830 I=1,NV
      DO 830 J=1,NV
830 AXTOT(I,J)=AXES(I,J)
      IF (IDIGIT(IOUTPUT,IOMAX-3).EQ.0) GO TO 850
      WRITE(06,*) ' CENTER OF GRAVITY OF ALLDATA '
      WRITE(06,70)(CGTOT(J),J=1,NV)
      WRITE(06,*) ' SEMIAXES OF BOTTLE CONTAINING ALLDATA (PRINTED
& IN ROWS) '
      DO 840 I=1,NAXTOT
      WRITE(06,70)(AXTOT(I,J)*AXLTOT(I),J=1,NV)
840 CONTINUE
      WRITE(06,*) ' LENGTH OF THE ALLDATA SEMIAXES '
      WRITE(06,70)(AXLTOT(I),I=1,NAXTOT)
850 IF (IDIGIT(IOUTPUT,IOMAX-4).EQ.0) GO TO 905
      WRITE(06,*) ' DATA TRANSFORM TO ALLDATA AXES (V) FROM ORIGINAL
& AXES (U) '
      DO 860 I=1,NAXTOT
      TEMP=0.
      DO 860 J=1,NV
860 TEMP=TEMP-AXTOT(I,J)*CGTOT(J)
      DO 870 J=1,NV
      IF (AXTOT(I,J).GE.0.)CHS(J)=CHP
      IF (AXTOT(I,J).LT.0.)CHS(J)=CHM
870 CONTINUE
      WRITE(06,100)CHV,1,TEMP,((CHS(J),ABS(AXTOT(I,J)),CHU,J),J=1,
&MINO(3,NV))
      IF (NV.GT.3)WRITE(06,110)((CHS(J),ABS(AXTOT(I,J)),CHU,J),
& J=4,NV)
880 CONTINUE
      WRITE(06,*) ' DATA TRANSFORM FROM ALLDATA AXES (V) TO ORIGINAL
& AXES (J) '
      DO 890 J=1,NV
      DO 890 I=1,NAXTOT
      IF (AXTOT(I,J).GE.0.)CHS(I)=CHP
      IF (AXTOT(I,J).LT.0.)CHS(I)=
&=CHM

```

```

890 CONTINUE
  WRITE(06,100)CHU,J,CGTOT(J),((CHS(I1),ABS(AXTOT(I1,J)),CHV,
    &I1),I1=1,MIND(3,NAXTOT))
  IF(NAXTOT.GT.3)WRITE(06,110)((CHS(I1),ABS(AXTOT(I1,J)),CHV,I1),
    &I1=4,NAXTOT)
900 CONTINUE
905 IF(IDIGIT(OUTPUT,IOMAX-5).EQ.0) GO TO 2085
  WRITE(06,*)
  WRITE(06,*) TRANSFORMED DATA POINTS, WITH POINT, SET NUMBER AT
  & BEGINNING OF ROW
  DO 910 I=1,NP
    WRITE(06,75)I,IPINSET(I),(XINF(I,J),J=1,NV)
910 CONTINUE
  GO TO 2085
C=====
930 DO 2000 I=1,ISETMAX
2000 NVINSET(I)=NPINSET(I)
  IF(IDIGIT(IOUTPS,IOMAXS).EQ.0) GO TO 931
  WRITE(06,*)
  WRITE(06,*) KNOWN VERTEX POINTS OF EACH SET
931 ITEMP0=0
  DO 2080 K=1,ISETNAM
    KK=(K-1)*4*NVMAX
    DO 2010 L=1,NV
      IF(AXLENG(L+(K-1)*NV).LT.AXLENG((1+(K-1)*NV)*.05) GO TO 2020
2010 CONTINUE
2020 CALL SORT(ITEMPMV(1+KK+2*NV),2*NV,1,0)
      J0=1
      DO 2025 I=1,NP
        IF(IABS(IPINSET(I)).NE.K) GO TO 2025
        ITEMP0=ITEMP0+1
        DO 2022 J=J0,2*NV
          IF(ITEMP0-ITEMPMV(J+KK+2*NV)) 2023,2021,2022
2021 ITEMPMV(J+KK+2*NV)=I
2022 CONTINUE
2023 JU=J
2025 CONTINUE
2029 CALL SORT(ITEMPMV(1+KK),4*NV,1,0)
      DO 2030 J=1,4*NV
        IF(ITEMPMV(J+KK).NE.0) GO TO 2040
2030 CONTINUE
2040 DO 2050 JJ=1,J
2050 ITEMPMV(JJ+KK)=ITEMPMV(J+KK)
      JJ=1
      DO 2060 J=2,4*NV
        IF(ITEMPMV(J+KK).EQ.ITEMPMV(J-1+KK)) GO TO 2060
        JJ=JJ+1
        ITEMPMV(JJ+KK)=ITEMPMV(J+KK)
2060 CONTINUE
      IF(JJ.EQ.0) GO TO 2075
      DO 2070 JJJ=JJ+1,J
2070 ITEMPMV(JJJ+KK)=0
2075 IF(IDIGIT(IOUTPS,IOMAXS).EQ.0) GO TO 2080
      WRITE(06,2076)K,JJ,NPINSET(K)
      WRITE(06,60)(ITEMPMV(JJJ+KK),JJJ=1,JJ)
2076 FORMAT(' SET NUMBER',14,' (',15,' OF',15,' POINTS)')
2080 CONTINUE
      GO TO 990
C-----
2085 IF(ISETNAM.LE.1) GO TO 6000

```



```

NV1=NV+1
L1=4*NVMAX*ISETNAM
L2=L1+4*NV1*(2+NV1)
L3=L2+4*NV1*NV1
L4=L3+NV1
L5=L4+NV1
L6=L5+NV1
L7=L6+4*NV1
L8=L7+NV1
IF (IDIGIT(IOUTPS,IOMAXS).NE.0)WRITE(06,*) ' REDUNDANT POINTS AS
& LINEAR COMBINATION OF VERTICES'
DO 2240 K=1, ISETNAM
KK=(K-1)*4*NVMAX
DO 2090 I=1, 4*NVMAX
IF (ITEMPMV(I+KK).EQ.0) GO TO 2100
2090 CONTINUE
I=I+1
2100 KVERTEX=I-1
IF (KVERTEX.EQ.NPINSET(K)) GO TO 2240
DO 2110 J=1, KVERTEX
TEMPMV(NV1+(J-1)*NV1+L2)=1.
DO 2110 I=1, NV
2110 TEMPMV(I+(J-1)*NV1+L2)=XINF(ITEMPMV(J+(K-1)*4*NVMAX), I,
CALL LUFACR(TEMPMV(1+L2), NV1, NV1, KVERTEX, ITEMPMV(1+L3),
& ITEMPMV(1+L4), ITEMPMV(1+L5), ITEMPMV(1+L6), IRANK)
CALL BACKS(TEMPMV(1+L2), NV1, NV1, KVERTEX, ITEMPMV(1+L3),
& ITEMPMV(1+L4), ITEMPMV(1+L5), ITEMPMV(1+L6), IRANK)
DO 2120 I=1, NV1*(4*NV1+7)
2120 TEMPMV(L1+I)=TEMPMV(L2+I)
I1=1
DO 2230 I=1, KVERTEX+1
I2=ITEMPMV(I+(K-1)*4*NVMAX)-1
IF (I.GT.KVERTEX) I2=NP
IF (I1.GT.I2) GO TO 2220
DO 2210 I1=I1, I2
IF (IPINSET(I1).NE.K) GO TO 2210
DO 2130 J=1, NV
2130 TEMPMV(J+L7)=XINF(I1, J)
TEMPMV(NV1+L7)=1.
DO 2140 J=1, NV1*(4*NV1+7)
2140 TEMPMV(L2+J)=TEMPMV(L1+J)
CALL NOSMPLX(TEMPMV(1+L2), NV1, NV1, KVERTEX, ITEMPMV(1+L3),
& ITEMPMV(1+L4), ITEMPMV(1+L5), ITEMPMV(1+L6), IRANK, TEMPMV(1+L7),
& IFEAS)
TEMPO=0.
DO 2150 J=1, NV
2150 TEMPMV(J+L5)=0.
DO 2170 I1=1, IRANK
I3=ITEMPMV(I1+L3)
I4=ITEMPMV(I1+L4)
I5=ITEMPMV(I4+(K-1)*
& 4*NVMAX)
IF (ITEMPMV(I3+L7).LE.0.) GO TO 2170
TEMPO=TEMPO+TEMPMV(I3+L7)
DO 2160 J=1, NV
2160 TEMPMV(J+L5)=TEMPMV(J+L5)+XINF(I5, J)*TEMPMV(I3+L7)
2170 CONTINUE
IF (IFEAS.EQ.0) GO TO 2190
TEMP1=0.
TEMP2=0.

```

```

DO 2180 J=1,NV
  TEMPI=TEMPI+(TEMPMV(J)*LS+TEMPO-XINF(II,J))*2
  TEMP2=TEMP2+XINF(II,J)*2
  IF(TEMP1.GT. TEMP2*10.E-8) GO TO 2210
  IPINSET(II)=-IPINSET(II)
  NVINSET(IABS(IPINSET(II)))=NINSET(IABS(IPINSET(II)))-1
  IF(IDIGIT(IOUTPS,IOMAXS).EQ.0) GO TO 2210
  DO 2195 I1=1,IRANK
    CHS(III)=CHP
    IF(TEMPMV(ITEMPMV(III)*L7)*L7).LT.0.)CHS(III)=CHM
  2195 CONTINUE
    WRITE(06,2200)II,(CHS(III),ABS(TEMPMV(ITEMPMV(III)*L3)+L7)),
    &ITEMPMV(ITEMPMV(III)*L4)*(K-1)*4*NVMAX,III=1,MINO(3,IRANK))
    I1=1
  2205 IF(4*I1-1.GE.IRANK) GO TO 2210
    WRITE(06,2207)(CHS(III),ABS(TEMPMV(ITEMPMV(III)*L3)+L7)),
    &ITEMPMV(ITEMPMV(III)*L4)*(K-1)*4*NVMAX,III=4*I1,MINO(4*I1+3,
    &IRANK))
  2207 FORMAT(4(2X,A1,F9.4,' P',I4))
    I1=I1+1
    GO TO 2205
  2210 CONTINUE
  2220 I1=12+2
  2230 CONTINUE
  2240 CONTINUE
    IF(IDIGIT(IOUTPS,IOMAXS).EQ.0) GO TO 2246
    ITEMPO=NP
    DO 2245 I=1,ISETNAM
      ITEMPO=ITEMPO-NVINSET(I)
    2245 WRITE(06,*) I,ROUND(ITEMPO, REDUNDANT POINTS OUT OF',NP,
    & POINTS
    -----
  2246 IF(IDIGIT(IOUTPS,IOMAXS-1).EQ.0) GO TO 2260
    WRITE(06,*)
    WRITE(06,*) NUMBER OF SET TO WHICH POINT BELONGS (-MEANS
    & REDUNDANT POINT)
    DO 2250 I=1,(NP+9)/10
      IPRINT=1+(I-1)*10
    2250 WRITE(06,80)(IPINSET(J),J=IPRINT,MINO(NP,IPRINT+9))
    CONTINUE
    -----
  2260 IF(IDIGIT(IOUTPS,IOMAXS-2).EQ.0.AND.IDIGIT(IOUTPS,IOMAXS-3).
    &EQ.0.AND.IDIGIT(IOUTPS,IOMAXS-10).EQ.0) GO TO 4000
    IER=0
    IER1=4
    CALL UERSET(1,IER,IER1)
    C SET P(NVP3,I) +, IF ITH PAIR OF SETS IS DISJOINT,OVERLAPS
    IF(IDIGIT(IOUTPS,IOMAXS-2).NE.0) WRITE(06,*) DISJOINTNESS OF
    & CONVEX HULLS OF PAIRS OF SETS, IN SPACE OF ALLDATA AXES'
    K=0
    DO 2340 I=1,ISETNAM-1
      DO 2340 J=I+1,ISETNAM
        I1=I
        JJ=J
        K=K+1
        P(NVP3,K)=0.
        DO 2270 L=1,NV
          IL=L
          CALL INSECT(XINF,IPINSET,I1,JJ,IL,NP,TEMPMV,ITEMPMV,IER)

```

```

IF(IER.EQ.133) GO TO 2330
2270 CONTINUE
IF(IER.NE.132) GO TO 2275
WRITE(06,2274)II,JJ,IL
2274 FORMAT(' SET ',I4,I4,' DISJOINTNESS UNDECIDED IN SPACE OF
& FIRST ',I4,' AXES')
2275 P(NVP3,K)=-IL
IF(IDIGIT(IOUTPS,IOMAXS-2).EQ.0) GO TO 2340
WRITE(06,2280)II,JJ,CHNOT,IL
2280 FORMAT(' SETS ',I4,I4,' ',A3,' DISJOINT IN SPACE OF FIRST ',I4
&,' AXES')
WRITE(06,*)' AN INTERSECTION IS:'
N1=2*(NVINSET(II)+NVINSET(JJ))+2*IL+7
N2=N1+IL+3
DO 2310 I1=1,ITEMPMV(N2)
CHS(I1)=CHP
IF(TEMPMV(N1+I1).LT.0.) CHS(I1)=CHM
2310 CONTINUE
WRITE(06,2320)(CHS(I1),ABS(TEMPMV(N1+I1)),ITEMPMV(N2+I1)),I1=1,
&MINO(4,ITEMPMV(N2)))
2320 FORMAT(' 0=',4(A1,F9.4,' P',I4))
I1=1
2325 IF(4*I1-1.GE.ITEMPMV(N2)) GO TO 2340
WRITE(06,2327)(CHS(I2),ABS(TEMPMV(N1+I2)),ITEMPMV(N2+I2),
&I2=4*I1+1,MINO(4*(I1+1),ITEMPMV(N2)))
2327 FORMAT(3X,4(A1,F9.4,' P',I4))
I1=I1+1
GO TO 2325
2330 IF(IDIGIT(IOUTPS,IOMAXS-2).NE.0) WRITE(06,2280)II,JJ,CHBL,IL
P(NVP3,K)=+IL
2340 CONTINUE
C-----
2370 IF(IDIGIT(IOUTPS,IOMAXS-3).EQ.0.AND.IDIGIT(IOUTPS,IOMAXS-10).
&EQ.0) GO TO 4000
DO 2380 I=1,NP
WRITE(22,ERR=2390)(XINF(I,J),J=1,NV)
2380 CONTINUE
REWIND 22
GO TO 2400
2390 WRITE(06,*)' ERROR IN WRITING XINF FILE'
STOP 8
C-----
2400 WRITE(06,*)'
WRITE(06,*)' HYPERPLANES FOR SET DISCRIMINATION, IN TERMS OF
& TRANSFORMED VARIABLES'
K=0
TEMP1=1.
DO 2440 I=1,ISETNAM-1
DO 2440 J=I+1,ISETNAM
I1=I
JJ=J
K=K+1
2506 ITEMPO=0
2402 SOS=+1.
IF(P(NVP3,K).GT.0.) SOS= 1.
L=ABS(1.001*P(NVP3,K))
CALL DISCRM (XINF,IPINSET,II,JJ,L,NP,NVINSET(II)+NVINSET(JJ)
&P(2,K),P(1,K),P(NVP3-1,K),SOS,IER)
IF(IER.NE.0) WRITE(06,*)' DISCRM IER =',IER,' FOR SETS',I,J
IF(IER.LT./0.A.10) P(NVP3,K).LT.0..AND.ITEMPO.EQ.1.AND.L.LT.NV)

```

```

& GO TO 2509
IF(IER.LT.70) GO TO 2408
IF(ITEMPO.NE.U) GO TO 2405
WRITE(06,2403)I,J
2403 FORMAT(' TRYING OPPOSITE CASE AS
& HYPERPLANES NOT SOLVABLE FOR SETS',I3,' ',I3)
P(NVP3,K)=-P(NVP3,K)
ITEMPO=1
GO TO 2402
2405 WRITE(06,*) ' NEITHER OVERLAP NOR SEPARATION CASE SOLVABLE FOR
& SETS',I,J
2509 P(NVP3,K)=-P(NVP3,K)
IF(L.GE.NV.AND.IER.EQ.132) GO TO 2408
IF(L.GE.NV) GO TO 2440
P(NVP3,K)=P(NVP3,K)+SIGN(ABS(TEMP1),P(NVP3,K))
WRITE(06,*) ' TRYING SPACE OF',L+1,' VARIABLES'
GO TO 2506
2408 WRITE(06,2410)I,P(1,K),J,P(NVP3-1,K)
2410 FORMAT(' HYPERPLANES FOR SETS',I2,' ',E11.4,' LE RHS)&',I2,
&(' ',E11.4,' GE RHS'))
DO 2420 I1=2,L+1
IF(P(I1,K).GE.U) CHS(I1)=CHP
IF(P(I1,K).LT.U) CHS(I1)=CHM
2420 CONTINUE
WRITE(06,2430)((CHS(J1),ABS(P(J1,K)),CHV,J1-1),
& J1=2,MINO(4,L+1))
IF(L+1.GT.4)WRITE(06,110)((CHS(J1),ABS(P(J1,K)),CHV,J1-1),
& L+1)
2430 FORMAT(' RHS IS',A2,E10.3,' ',A1,' ',I2,' ',A2,
& E10.3,' ',A1,' ',I2,' ',A2,E10.3,' ',A1,' ',I2,' ')
2440 CONTINUE
WRITE(06,*) '
K=0
DO 2445 I=1,ISETNAM-1
DO 2445 J=I+1,ISETNAM
K=K+1
IF(P(NVP3,K).GE.U) WRITE(06,2446)I,J,P(1,K)-P(NVP3-1,K)
IF(P(NVP3,K).LT.U) WRITE(06,2447)I,J,P(NVP3-1,K)-P(1,K)
2445 CONTINUE
2446 FORMAT(' SEPARATION BETWEEN SETS',I3,' &',I3,' = ',E11.4)
2447 FORMAT(' OVERLAP OF SETS',I3,' &',I3,' = ',E11.4)
C=====
WRITE(06,*) '
WRITE(06,*) ' SAMPLE PLANES APPLIED TO TRANSFORMED SAMPLE
& FILES, LESS EXCLUSIONS'
WRITE(06,*) ' POINT SET DISTANCE(RHS) TO POINT, NORMAL TO
& PLANES (E=ERROR,B=BAND)'
ITEMPO=(ISETNAM*(ISETNAM-1))/2
WRITE(06,2450)((I,J,I+1,ISETNAM),I=1,ISETNAM-1)
2450 FORMAT(' NO. ID NO.',I3,I2,7(I6,I2):/,
& 50(' ',B(16,I2):/))
TOIN=1E-5
TOLRED=.2
DO 2455 I=1,NSPAIR
HCONL(I)=V3IGNO
HCONR(I)=-V8IGNO
TEMPV(2*NSPAIR+1)=0.

```

```

2455 TEMPV(NSPAIR*I)=0.
DO 2500 I=1,N
  READ(22)(BUFFER(I),J1=1,NV)
  ID=IABS(IPINSET(I))
  K1=0
  ITEMP=0
DO 2470 J1=1,ISETNAM-1
DO 2470 J2=J1+1,ISETNAM
  K1=K1+1
  TEMPV(K1)=0
  L=ABS(1.001*P(NVP3,K1))
DO 2460 L=1,L
  CHEB(K1)=TEMPV(K1)+BUFFER(L1)*P(L1+1,K1)
  CHEB(K1)=CHBL1
  IF(I1.NE.ID.AND.J1.NE.ID) GO TO 2470
  IF(I1.EQ.ID.AND.TEMPV(K1)-P(1,K1).LT.-TOLH)CHEB(K1)=CHE
  IF(J1.EQ.ID.AND.TEMPV(K1)-P(NVP3-1,K1).GT.+TOLH)CHEB(K1)=CHE
  IF(TEMPV(K1)-P(1,K1).GE.-TOLH.AND.TEMPV(K1)-P(NVP3-1,K1).LE.
    &+TOLH)
    &CHEB(K1)=CHB
  IF(I1.EQ.ID) TEMP1=TEMPV(K1)-TOLRED
  IF(J1.EQ.ID) TEMP1=TEMPV(K1)+TOLRED
  IF(P(NVP3,K1).GE.0.AND.I1.EQ.ID.AND.TEMP1.LT.P(1,K1)) ITEMP=1
  IF(P(NVP3,K1).LT.0.AND.I1.EQ.ID.AND.TEMP1.LT.P(NVP3-1,K1))
    &ITEMP=1
  IF(P(NVP3,K1).GE.0..AND.J1.EQ.ID.AND.TEMP1.GT.P(NVP3-1,K1))
    &ITEMP=1
  IF(P(NVP3,K1).LT.0..AND.J1.EQ.ID.AND.TEMP1.GT.P(1,K1))ITEMP=1
  IF(CHEB(K1).EQ.CHB) TEMPV(2*NSPAIR+K1)=TEMPV(2*NSPAIR+K1)+1.
  IF(CHEB(K1).EQ.CHE) TEMPV(NSPAIR+K1)=TEMPV(NSPAIR+K1)+1.
  IF(I1.EQ.ID.AND.I.GT.1) HCONL(K1)=AMIN1(HCONL(K1),TEMPV(K1))
  IF(J1.EQ.ID.AND.I.GT.1) HCONR(K1)=AMAX1(HCONR(K1),TEMPV(K1))
2470 CONTINUE
  IF(ITEMP.EQ.0) IPINSET(I)=-IABS(IPINSET(I))
DO 2475 K2=1,ITEMP
  IF(ABS(TEMPV(K2)).GT.9.9999) TEMPV(K2)=SIGN(9.9999,TEMPV(K2))
2475 CONTINUE
  WRITE(06,2480)I,ISETID(ID),IPINSET(I),(TEMPV(K2),CHEB(K2),
    &K2=1,MINO(ITEMP0,B))
  IF(ITEMP0.GT.B)WRITE(06,2490)(TEMPV(K2),CHEB(K2),K2=9,ITEMP0)
2480 FORMAT(14,12,12,8,(F7.4,A1))
2490 FORMAT(50(
    ,B(F7.4,A1):/))
2500 CONTINUE
REWIND 22
DO 2501 I=1,NSPAIR
  IF(TEMPV(NSPAIR*I).NE.0.) GO TO 2502
2501 CONTINUE
  WRITE(06,*)
  WRITE(06,*) NO E'S OCCURRED. (NO VIOLATION OF HYPERPLANES)
  GO TO 2507
2502 WRITE(06,*)
  WRITE(06,2503)(TEMPV(NSPAIR*K2),K2=1,MINO(ITEMP0,B))
  IF(ITEMP0.GT.B) WRITE(06,2504)(TEMPV(NSPAIR*K2),K2=9,ITEMP0)
2503 FORMAT(' E TOTALS',F6.0,(F8.0))
2504 FORMAT(50(
    ,B(F8.0):/))
2507 WRITE(06,*)
DO 2511 I=1,NSPAIR
  IF(TEMPV(2*NSPAIR-I).NE.0.) GO TO 2512
2511 CONTINUE

```

```

WRITE(06,*) ' NO B'S OCCURRED. (NO PTS IN HYPERPLANE OVERLAP)'
GO TO 2517

2512 WRITE(06,2513)(TEMPV(2*NSPAIR+K2),K2=1,MINO(ITEMPO,8))
IF(ITEMPO.GT.8)WRITE(06,2504)(TEMPV(2*NSPAIR+K2),K2=9,ITEMPO)
2513 FORMAT(' B TOTALS',F6.0,7(F8.0))

2517 WRITE(06,*) '
WRITE(06,*) ' MINIMUM OF HYPERPLANE RHS FOR LEFT SETS'
WRITE(06,2490)(HCONL(K2),CHBL1,K2=1,ITEMPO)
WRITE(06,*) ' MAXIMUM OF HYPERPLANE RHS FOR RIGHT SETS'
WRITE(06,2490)(HCONR(K2),CHBL1,K2=1,ITEMPO)
WRITE(06,*) ' IMPROVEMENTS TO HYPERPLANE LEFT HAND SIDES'
WRITE(06,*) ' LEFTNO, RIGHTNO, LEFTOLD, RIGHTOLD, LEFTNEW, RIGHTNEW'
5 RIGHTNEW
K1=0
DO 2515 I1=1,ISETNAM-1
DO 2515 J1=I1+1,ISETNAM
K1=K1+1
K2=0
IF(P(NVP3,K1).GE.0.AND.P(NVP3-1,K1).LT.P(1,K1).AND.
&HCONR(K1).LT.HCONL(K1)) K2=1
IF(P(NVP3,K1).LT.0.AND.P(1,K1).LT.P(NVP3-1,K1).AND.
&HCONL(K1).GT.P(1,K1)-TOLH.AND.HCONR(K1).LT.P(NVP3-1,K1)
&TOLH) K2=1
IF(K2.EQ.0) GO TO 2515
WRITE(06,2516) I1,J1,P(1,K1),P(NVP3-1,K1),HCONL(K1),HCONR(K1)
2516 FORMAT('16,17',,4E11.4)
P(1,K1)=HCONL(K1)
P(NVP3-1,K1)=HCONR(K1)
2515 CONTINUE

2520 IF(1DIGIT(IOUTPS,IOMAXS-1).EQ.0) GO TO 3000
WRITE(06,*) '
WRITE(06,*) ' NUMBER OF SET TO WHICH POINT BELONGS (-MEANS
& REDUNDANT POINT)'
DO 2510 I=1,(NP+9)/10
IPRINT=1+(I-1)*10
WRITE(06,80)(IPINSET(J),J=IPRINT,MINO(NP,IPRINT+9))
2510 CONTINUE
=====
3000 IF(1DIGIT(IOUTPS,IOMAXS-10).EQ.0) GO TO 4000
WRITE(06,*) '
WRITE(06,*) ' NAME REVISED SAMPLE & POPULATION FILES (IN SAMPLE
& FORMAT(' , , IF NONE)
READ(05,*)FILINR,FILINT
IF(IMODE.EQ.0)WRITE(06,*)FILINR,FILINT
IF(FILINR.EQ.FILECR.AND.FILINT.EQ.FILECR) GO TO 4000
IF(FILINT.EQ.FILECR) GO TO 3049
NSETOUTT=0
DO 3051 I=1,ISETMAX
3051 ISETOUTT(I)=0
WRITE(06,3059)ISETMAX
3059 FORMAT(' ENTER NLE,14,') THEN ID OF N SETS TO BE EXCLUDED
& (-1 FOR ALL NEW SETS)
READ(05,*)NSETOUTT,(ISETOUTT(I),I=1,MINO(NSETOUTT,ISETMAX))
IF(IMODE.EQ.0)
34WRITE(06,*)NSETOUTT,(ISETOUTT(I),I=1,MINO(NSETOUTT,ISETMAX))
WRITE(06,*) ' ENTER NLE 20) THEN NAME N POPULATION POINTS TO BE
& EXCLUDED.
READ(05,*)NP1OUTT1,(IPTOUTT(I),I=1,MINO(NP1OUTT,20))

```

```

IF (IMODE.EQ.0)
&WRITE(06,*INPTOUTT,(IPTCOT(I),I=1,MINO(NPTOUTT,20))
INW=0
IF (NSETCOT.EQ.0) GO TO 3054
DO 3053 I=1,NSETCOT
IF (ISETCOT(I).EQ.-1) GO TO 3052
3053 CONTINUE
GO TO 3054
3052 INW=1
3054 WRITE(06,*) ' ENTER N, THEN N PAIRS OF IDS TO BE LABELED AS THE
& FIRST'
READ(05,*)NSETCOT,(ISETCOT(I),I=1,2*MINO(NSETCOT,ISETMAX))
IF (IMODE.EQ.0)
&WRITE(06,*INSETCOT,(ISETCOT(I),I=1,2*MINO(NSETCOT,ISETMAX))
C *****THE FOLLOWING IS OBSOLETE AND COMMENTED OUT
C 3049 IF (FILNR.EQ.FILECR) GO TO 3064
C CALL CONCAT(FILENAME,1,FILNR,1,8)
C CALL PCREAT(FILENAME,(NPMAX*INMXMX*5*319)/320,3,0,ISTAT,TEMPW)
C IF (ISTAT.EQ.0) GO TO 3055
C WRITE(06,3057)FILENAME,ISTAT
C 3057 FORMAT(' PCREAT FOR ',A9,' RESULTS IN ISTAT=',O12)
C IF (IMODE.EQ.1) GO TO 3061
C CALL PTIME(TIMEOUT)
C WRITE(FILNR,3063)(TIMEOUT-TIMEIN)*3600.
C 3063 FORMAT(F8.2)
C WRITE(06,*) ' NAMING REVISED SAMPLE FILE ',FILNR
C GO TO 3062
C 3061 WRITE(06,*) ' NAME REVISED SAMPLE FILE (' ' ' IF NONE)'
C READ(05,*)FILNR
C IF (IMODE.EQ.0)WRITE(06,*)FILNR
C 3062 IF (FILNR.EQ.FILECR.AND.FILINT.EQ.FILECR) GO TO 4000
C GO TO 3049
C 3055 CALL ATTACH(03,FILENAME,3,0,ISTAT,)
C IF (ISTAT.NE.1,ISTATOK.AND.1,ISTAT.NE.1,ISTAT2) WRITE(06,240)ISTAT
C 240 FORMAT(' ERROR. ATTACH STATUS IS ',O12)
C *****END OF OBSOLETE SECTION
C *****START OF NEW SECTION
3049 IF (FILNR.EQ.FILECR) GO TO 3064
INQUIRE(FILE=FILNR,EXIST=EXIS)
IF (.NOT.EXIS) THEN
& OPEN(03,FILE=FILNR,STATUS='NEW',Iostat=ISTAT,
& ACCESS='SEQUENTIAL',FORM='FORMATTED')
& WRITE(06,*)
& WRITE(06,*) ' ERROR OPENING REVISED SAMPLE FILE. ISTAT=',ISTAT
& STOP
& END IF
& IF (EXIS) THEN
& IF (IMODE.EQ.1) THEN
& WRITE(06,*) ' REVISED SAMPLE FILE ',FILNR, ' ALREADY EXISTS.'
& WRITE(06,*) ' ENTER 1 TO OVERWRITE OR 2 TO ENTER NEW NAME.'
& READ(05,*) IOVERW
& IF (IOVERW.NE.1 .AND. IOVERW.NE.2) THEN
& WRITE(06,*) ' MUST ENTER 1 OR 2. PLEASE TRY AGAIN.'
& GO TO 3050
& END IF
& IF (IOVERW.EQ.1) THEN
& OPEN(03,FILE=FILNR,STATUS='OLD',Iostat=ISTAT,
& ACCESS='SEQUENTIAL',FORM='FORMATTED')
&

```

```

      IF (ISTAT.NE.0) THEN
        WRITE(06,*) 'ERROR OPENING REVISED SAMPLE FILE.',
          , ISTAT, ISTAT
        STOP
      END IF
    END IF
  END IF
  IF (IOVERW.EQ.0) THEN
    WRITE(06,*) 'ENTER NAME OF REVISED SAMPLE FILE.',
      , (0) IF NONE)
    READ(05,*) FILINR
    IF (FILINR.EQ.FILECR) .AND. FILINT.EQ.FILECR) GO TO 4000
    GO TO 3049
  END IF
END IF
END IF
IF (IMODE.EQ.0) THEN
  IF (TIME(TIMEOUT)
    CALL PTIME(TIMEOUT)
    WRITE(FILINR,3063)INT((TIMEOUT-TIMEIN)*3600.*1000.)
    FORMAT(I8)
    WRITE(06,*) 'NAMING REVISED SAMPLE FILE',FILINR
    OPEN(03,FILE=FILINR,STATUS='NEW',IOSTAT=ISTAT,
      , ACCESS='SEQUENTIAL',FORM='FORMATTED')
    IF (ISTAT.NE.0) THEN
      WRITE(06,*) 'ERROR OPENING REVISED SAMPLE FILE.',
        , ISTAT, ISTAT
      STOP
    END IF
  END IF
END IF
END IF
END IF
C*****END OF NEW SECTION
CALL FMEDIA(03,6)
3064 ITEM1=0
ITEM2=0
ITEM3=0
ITEM4=0
ITEM5=0
ITEM6=0
ITEM7=0
ITEM8=0
K3=NSPAIR*NVNLMAX
DO 3069 I=1,6*ISETM2
  IF(FILINR.EQ.FILECR) GO TO 3080
DO 3058 I=1, ISETM2
  NPINSET(I)=0
  NFILE=1
  CALL ATFILE(FILIN(I),NFILE)
  I=0
  NPTMP=0
3070 I=I+1
3071 READ(01,90,END=3075)(BUFFER(L),L=1,INMAX)
  NPTMP=NPTMP+1
  ITEM=BUFFER(ISETVAR)
  IF(NSETOUT.EQ.0.AND.NPTOUT.EQ.0.AND.NSETCOM.EQ.0) GO TO 3079
  IF(NSETOUT.LE.0) GO TO 3073
  DO 3072 K=1,NSETOUT
    IF(ITEMP.EQ.ISETOUT(K)) GO TO 3071
3072 CONTINUE
3073 IF(NPTOUT.LE.0) GO TO 3076
DO 3074 K=1,NPTOUT

```



```

IF(NPTMP.EQ.IPTOUT(K)) GO TO 3071
3074 CONTINUE
3076 IF(NSETCOM.LE.0) GO TO 3079
DO 3077 K=1,NSETCOM
IF(IITEMP.EQ.ISETCOM(2*K)) BUFFER(ISETVAR)=ISETCOM(2*K-1)
3077 CONTINUE
3079 IF(IPINSET(1).LT.0.AND.I.LE.NPMAX) GO TO 3070
ITEMPI=ITEMPI+1
NPINSET(IABS(IPINSET(1)))=NPINSET(IABS(IPINSET(1)))+1
WRITE(U3,90,ERR=3060)(BUFFER(L),L=1,INMAX)
GO TO 3070
3060 WRITE(06,*) ' ERROR IN TRANSCRIBING PT',I,' FROM SAMPLE TO
& REVISED SAMPLE DATA BASE. PROCEEDING.'
GO TO 3070
C
3075 NFILE=NFILE+1
CALL ATFILE(FILIN,NFILE)
IF(NFILE.NE.0) GO TO 3070
3080 ISETNAM=ISETNAM
IF(FILINT.EQ.FILECR) GO TO 3499
C
C=====
WRITE(06,*)
WRITE(06,*) ' SAMPLE PLANES APPLIED TO TRANSFORMED POP FILES
& LESS NEW EXCLUSIONS'
WRITE(06,*) ' POINT SET DISTANCE(RHS) TO POINT, NORMAL 10
& PLANES (E=ERROR,B=BAND)
IO=(ISETNAM*(ISETNAM-1))/2
I2=MAX(10,NV)
WRITE(06,2450)((I,J,J=I+1,ISETNAM),I=1,ISETNAM-1)
NPT=0
DO 3085 I=1,NSPAIR
3085 TEMPV(NSPAIR+NVMAX*I)=0.
3100 NPTMP=0
OPEN(U2,FILE=FILINT,STATUS='OLD',Iostat=Istat,
&ACCESS='SEQUENTIAL',FORM='FORMATTED')
IF (Istat.NE.0) THEN
WRITE(06,*) ' ERROR OPENING FILINT FILE. ISTAT= ',Istat
STOP
END IF
3220 READ(02,90,END=3495)(BUFFER(L),L=1,INMAX)
NPTMP=NPTMP+1
ITEMP=BUFFER(ISETVAR)
IF(NSETOUTT.EQ.0.AND.NPTOUTT.EQ.0.AND.NSETCOT.EQ.0) GO TO 3235
IF(NSETOUTT.LE.0) GO TO 3225
DO 3224 K=1,NSETOUTT
IF(IITEMP.EQ.IPTOUT(K)) GO TO 3220
3224 CONTINUE
IF(INEW.EQ.0) GO TO 3225
DO 3221 K=1,ISETNAM
IF(IITEMP.EQ.ISETID(K)) GO TO 3225
3221 CONTINUE
GO TO 3220
3225 IF(NPTOUT.LE.0) GO TO 3235
DO 3230 K=1,NPTOUTT
IF(NPTMP.EQ.IPTOUTT(K)) GO TO 3220
3230 CONTINUE
3235 IF(NSETCOT.EQ.0) GO TO 3239
DO 3233 K=1,NSETCOT

```

```

C
IF (ITEMP.EQ. ISETCOT(2*K)) ITEMV=ISETCOT(2*K-1)

3233 CONTINUE
3239 IPINST=ISETID(1)
IF (ISETVAR.GT.0) IPINST=ITEMV
DO 3240 ID=1, ISETNAM
IF (IPINST.EQ. ISETID(ID)) GO TO 3250
3240 CONTINUE
IF (ISETNAMP.EQ.0) GO TO 3245
DO 3244 ID=ISETNAM+1, ISETNAMP
IF (IPINST.EQ. ISETID(ID)) GO TO 3247
3244 CONTINUE
3245 ISETNAMP=ISETNAM+1
ISETID(ISETNAMP)=IPINST
ID=ISETNAMP
3247 NPINSETT(ID)=NPINSETT(ID)+1
ITEMP3=ITEMP3+1
WRITE(03,90,ERR=3260)(BUFFER(L),L=1,INMAX)
GO TO 3251
3260 WRITE(06,*) ' ERROR IN WRITING POPULATION PT. NPTEMP, ' ONTO
& REVISED SAMPLE DATA BASE. PROCEEDING.'
3250 NPT=NPT+1
3251 DO 3300 J=1,NV
3300 TEMPV(J)=(BUFFER(IN(J))-STYP(J))/SVAR(J)-CGTOT(J)
DO 3320 I=1,NV
TEMPV(I+I2)=0.
DO 3320 J=1,NV
3320 TEMPV(I+I2)=TEMPV(I+I2)+AXTOT(I,J)*TEMPV(J)
C
K1=0
CHCF=CHBL1
DO 3470 I1=1, ISETNAM-1
DO 3470 J1=I1+1, ISETNAM
K1=K1+1
TEMPV(K1)=0.
L=ABS(1.001*P(NVP3,K1))
DO 3460 L1=1,L
TEMPV(K1)=TEMPV(K1)+P(L1+1,K1)*TEMPV(L1+I2)
CHEB(K1)=CHBL1
IF (I1.NE.ID.AND. J1.NE.ID) GO TO 3465
IF (I1.EQ.ID.AND. TEMPV(K1)-P(1,K1).LT.-TOLH) CHCF=CHE
IF (J1.EQ.ID.AND. TEMPV(K1)-P(NVP3-1,K1).GT.+TOLH) CHCF=CHE
IF (I1.EQ.ID) TEMP1=TEMPV(K1)-TOLRED
IF (J1.EQ.ID) TEMP1=TEMPV(K1)+TOLRED
1465 TEMP2=AMAX1(P(1,K1),P(NVP3-1,K1))
TEMP3=AMIN1(P(1,K1),P(NVP3-1,K1))
IF (TEMPV(K1).GT.TEMP2) CHEB(K1)=CHGT
IF (TEMPV(K1).LT.TEMP3) CHEB(K1)=CHLT
IF (TEMPV(K1).LE.P(1,K1).AND. TEMPV(K1).GE.P(NVP3-1,K1)) CHEB(K1)
& = CHEQ
IF (TEMPV(K1).GE.P(1,K1).AND. TEMPV(K1).LE.P(NVP3-1,K1)) CHEB(K1)
& = CHNO
3470 CONTINUE
IF (CHCF.EQ.CHE) TEMPV(NSPAIR+NVMAX+ID)=TEMPV(NSPAIR+NVMAX-ID)
& +1
IF (CHCF.NE.CHE) GO TO 3489
IF (ID.GT.ISETNAM) GO TO 3489
DO 3476 L=1, ISETNAM
ITEMPV(L)=0

```

```

IF(L.EQ.ISETNAM) GO TO 3475
K1=((2*ISETNAM-1)*L-1)/2
DO 3474 J=L+1,ISETNAM
IF(CHEB(K1+J).EQ.CHF) GO TO 3478
3474 CONTINUE
3475 IF(L.EQ.1) GO TO 3477
DO 3476 I=1,L-1
K1=((2*ISETNAM-1)*(I-1))/2-1
IF(CHEB(K1).EQ.CHF) GO TO 3478
3476 CONTINUE
3477 ITEMPV(L)=1
3478 CONTINUE
ITEMP=0
DO 3479 K1=1,ISETNAM
ITEMP=ITEMP+ITEMPV(K1)
IF(ITEMPV(ID).EQ.0.OR.ITEMP.NE.1) GO TO 3461
ITEMP4=ITEMP+1
ITEMPMV(K3+ID)=ITEMPMV(K3+ID)+1
3461 IF(ITEMPV(ID).EQ.0.OR.ITEMP.LE.1) GO TO 3462
ITEMP5=ITEMP+1
ITEMPMV(K3+ISETNAM)=ITEMPMV(K3+ISETNAM+ID)+1
3462 IF(ITEMP.NE.0) GO TO 3463
ITEMP6=ITEMP+1
ITEMPMV(K3+ISETNAM*2+ID)=ITEMPMV(K3+ISETNAM*2+ID)+1
3463 IF(ITEMPMV(ID).NE.0.OR.ITEMP.LE.1) GO TO 3464
ITEMP7=ITEMP+1
ITEMPMV(K3+ISETNAM*3+ID)=ITEMPMV(K3+ISETNAM*3+ID)+1
3464 IF(ITEMPV(ID).NE.0.OR.ITEMP.NE.1) GO TO 3488
ITEMP=0
IF(ID.EQ.ISETNAM) GO TO 3482
K1=((2*ISETNAM-ID)*(ID-1))/2-ID
DO 3481 I=ID+1,ISETNAM
IF(CHEB(K1+J).EQ.CHEQ) GO TO 3486
3481 CONTINUE
3482 IF(ID.EQ.1) GO TO 3485
DO 3483 I=1,ID-1
K1=((2*ISETNAM-I)*(I-1))/2-ID-I
IF(CHEB(K1).EQ.CHEQ) GO TO 3486
3483 CONTINUE
3485 ITEMPO=1
3486 IF(ITEMPO.NE.0) GO TO 3484
ITEMP8=ITEMP+1
ITEMPMV(K3+ISETNAM*4+ID)=ITEMPMV(K3+ISETNAM*4+ID)+1
3484 IF(ITEMPO.EQ.0) GO TO 3488
ITEMP9=ITEMP+1
ITEMPMV(K3+ISETNAM*5+ID)=ITEMPMV(K3+ISETNAM*5+ID)+1
3488 NPINSETT(IABS(ID))=NPINSETT(IABS(ID))+1
WRITE(U3,90,ERR=3480)(BUFFER(L),L=1,INMAX)
ITEMP2=ITEMP2+1
GO TO 3490
3460 WRITE(06,*) ' ERROR IN WRITING POPULATION PT',NPTEMP,' ONTO
      & REVISED SAMPLE DATA BASE. PROCEEDING.'
3490 IF(ITEMP.EQ.0) ID=-ID
DO 3492 K2=1,I0
IF(IABS(ITEMPV(K2)).GT.9.9999) TEMPV(K2)=SIGN(9.9999,TEMPV(K2))
3492 CONTINUE
WRITE(06,3487)NPTEMP,IPINSET,CHCF,IABS(ID),(TEMPV(K2),
&CHEB(K2),K2-1,MING(I0,B))

```

```

IF (IO.GT.8) WRITE(06,2490)(TEMPV(K2),CHEB(K2),K2=9,10)
3487 FORMAT(14,12,A),1),6(F7.4,A1))
3494 GO TO 3220

3495 REWIND 02
CLOSE(02,STATUS='KEEP')
WRITE(06,*)
WRITE(06,*) ENTER NAME OF NEXT POPULATION FILE (' ' IF NONE)
READ(05,*)FILINT
IF (IMODE.EQ.0) WRITE(06,*)FILINT
IF (FILINT.EQ.FILECR) GO TO 3499
NSETOUTT=0
DO 3491 I=1,ISETMAX
3491 ISETOUTT(I)=0
WRITE(06,*) ENTER N (LE I, ISETMAX,) THEN ID OF N SETS TO BE
& EXCLUDED.
READ(05,*)NSETOUTT,(ISETOUTT(I),I=1,MIN0(NSETOUTT,ISETMAX))
IF (IMODE.EQ.0)
&WRITE(06,*)NSETOUTT,(ISETOUTT(I),I=1,MIN0(NSETOUTT,ISETMAX))
WRITE(06,*) ENTER N (LE 20) THEN NAME N POPULATION POINTS TO BE
& EXCLUDED.
READ(05,*)NPTOUTT,(IPTOUTT(I),I=1,MIN0(NPTOUTT,20))
IF (IMODE.EQ.0)
&WRITE(06,*)NPTOUTT,(IPTOUTT(I),I=1,MIN0(NPTOUTT,20))
GO TO 3100
3499 REWIND 03
DO 3497 I=1,NSPAIR
IF (TEMPV(NSPAIR+VMAX+1).NE.0.) GO TO 3496
3497 CONTINUE
GO TO 3498
3496 WRITE(06,*)
WRITE(06,2503)(TEMPV(NSPAIR+VMAX+K2),K2=1,MIN0(10,8))
IF (IO.GT.8) WRITE(06,2504)(TEMPV(NSPAIR+VMAX+K2),K2=9,10)
3498 WRITE(06,*)
WRITE(06,*) .ITEMP1+ITEMP2+ITEMP3, POINTS ON REVISED SAMPLE
& FILE
WRITE(06,*) .ITEMP1, NON REDUNDANT FROM INPUT SAMPLE FILE
WRITE(06,*) .ITEMP3, FROM POPLATN FILES, WITH ID NOT IN
& SAMPLE FILE
WRITE(06,*) .NPT, FROM POPLATN FILES, WITH ID IN SAMPLE
& SAMPLE FILE
WRITE(06,*) .NPT-ITEMP2, FROM POPLATN FILES, CORRECTLY
& CLASSIFIED
WRITE(06,*) .ITEMP2, FROM POPLATN FILE, NOT CORRECTLY
& CLASSIFIED
WRITE(06,*) .ITEMP4, CORRECTABLE, UNIQUE, USING
& GAPS
WRITE(06,3501)(ITEMPMV(K3+K2),K2=1,ISETNAM)
WRITE(06,*) .ITEMP5, CORRECTABLE, NOT UNIQUE, USING
& GAPS
WRITE(06,3501)(ITEMPMV(K3+ISETNAM+K2),K2=1,ISETNAM)
WRITE(06,*) .ITEMP6, NO CLASSIFICATION, USING GAPS
WRITE(06,3501)(ITEMPMV(K3+ISETNAM+K2),K2=1,ISETNAM)
WRITE(06,*) .ITEMP7, INCORRECT CLASS, NOT UNIQUE
& USING GAPS
WRITE(06,3501)(ITEMPMV(K3+ISETNAM+K2),K2=1,ISETNAM)
WRITE(06,*) .ITEMP8, INCORRECT CLASS, UNIQUE, USING
& GAPS
WRITE(06,3501)(ITEMPMV(K3+ISETNAM+K2),K2=1,ISETNAM)
WRITE(06,*) .ITEMP9, INCORRECT CLASS, UNIQUE.

```

```

* WITHOUT GAPS
WRITE(06,3501)((ITEMPMV(K3+ISETNAM*S+K2),K2=1,ISETNAM)
3501 FORMAT(
PER SET:',915)
WRITE(06,*)
WRITE(06,3500)FILNR,ISETNAM,ISETNAM-ISETNAM
3500 FORMAT(' POINTS ON REVISED SAMPLE FILE ',A8,' (',I4,' OLD
& SETS, ',I4,' NEW SETS)')
WRITE(06,3493) SET ID',(ISETID(I),I=1,ISETNAM)
WRITE(06,3493) POINTS PER SET',(NPINSET(I),I=1,ISETNAM)
3493 FORMAT(A16,916)
GO TO 4000
C =====
4000 NSPAR=ISETNAM*(ISETNAM-1)/2
DO 4030 K=1,NSPAR
DO 4030 J=1,NV
TEMPM(K,J)=0
DO 4020 L=1,ABS(1.001*P(NVP3,K))
4020 TEMPM(K,J)=TEMPM(K,J)+P(L+1,K)*AXTOT(L,J)
4030 CONTINUE
DO 4035 K=1,NSPAR
TEMPV(K)=0
DO 4035 J=1,NV
4035 TEMPV(K)=TEMPV(K)+TEMPM(K,J)*((-STYP(J)/SVAR(J)-CGTOT(J))
DO 4040 J=1,NV
DO 4040 K=1,NSPAR
4040 TEMPM(K,J)=TEMPM(K,J)/SVAR(J)
WRITE(06,*) HYPERPLANES FOR SET DISCRIMINATION, IN TERMS OF
& INPUT VARIABLES
K=0
DO 4060 I=1,ISETNAM-1
DO 4060 J=I+1,ISETNAM
K=K+1
WRITE(06,2410)I,P(1,K)-TEMPV(K),J,P(NVP3-1,K)-TEMPV(K)
DO 4050 I=1,NV
IF(TEMPM(K,I).GE.0.) CHS(I1)=CHP
IF(TEMPM(K,I).LT.0.) CHS(I1)=CHM
4050 CONTINUE
WRITE(06,2430)(CHS(J1),ABS(TEMPM(K,J1)),CHV,J1,J1=1,MIN0(3,NV))
IF(NV.GT.3) WRITE(06,110)(CHS(J1),ABS(TEMPM(K,J1)),CHV,J1,
& J1=4,NV)
4060 CONTINUE
C =====
C
C CHECK, REPRODUCE PLANE/PT DISTANCE USING PLANES IN INPUT VARIABLES
IF(10DIGIT(1000*PS,10MAXS-11).EQ.0) GO TO 4100
WRITE(06,*)
WRITE(06,*) SAMPLE PLANES APPLIED TO ORIGINAL SAMPLE
& FILES, NO EXCLUSIONS
WRITE(06,*) (USES ORIG PT NUMBERS, NOT INCLUDED PT
& NUMBERS
WRITE(06,*) POINT SET DISTANCE(RHS) TO POINT, NORMAL TO
& PLANES (E=ERROR, B=BAND)
WRITE(06,2450)((I,J,J=I+1,ISETNAM),I=1,ISETNAM-1)
NFILE=1
CALL ATFILE(FILIN,NFILE)
I=0
4065 READ(01,90,END=4067)(BUFFER(L),L=1,INMAX)
K1=0

```

```

ID=BUFFER(ISETVAR)
I=I+1
I4=0
DO 4066 I3=1,ISETNAM
  IF(ISETID(I3).EQ.ID) I4=I3
4066 CONTINUE
DO 4070 I1=1,ISETNAM-1
DO 4070 I2=I1+1,ISETNAM
  K1=K1+1
  TEMPV(K1+NSPAR)=0.
DO 4070 J=1,NV
  TEMPV(K1+NSPAR)=TEMPV(K1+NSPAR)+TEMPM(K1,J)*BUFFER(IN(J))
  WRITE(06,2480)I,I4,I4,(TEMPV(K+NSPAR),CHBL1,
    & K=1,MINO(NSPAR,B))
  IF(NSPAR.GT.B) WRITE(06,2490)(TEMPV(K+NSPAR),CHBL1,K=9,NSPAR)
  GO TO 4065
4067 NFILE=NFILE+1
  CALL ATFILE(FILIN,NFILE)
  IF(NFILE.NE.0) GO TO 4065
C=====
C WRITE HYPERPLANES ON FILEHO IN FORMAT:
C ISETID(I),ISETID(J),CONSTANTS FOR PAIR OF PLANES, COEFFICIENTS
4100 WRITE(06,*)
  READ(05,*)FILEHI,FILEHO
  IF(IMODE.EQ.0)WRITE(06,*)FILEHI,FILEHO
  IF(FILEHI.EQ.FILECR) GO TO 6000
  IF(FILEHI.EQ.FILECR) GO TO 4110
  OPEN(04,FILE=FILEHI,STATUS='OLD',IOSTAT=ISTAT,
    & ACCESS='SEQUENTIAL',FORM='FORMATTED')
  IF (ISTAT.NE.0) THEN
    WRITE(06,*) ' ERROR OPENING INPUT HYPERPLANE FILE. ISTAT=',
      ISTAT
    STOP
  END IF
C *****OBsolete CODING COMMENTED OUT
C 4110 CALL CONCAT(FILENAME,1,FILEHO,1,8)
C 4110 CALL PCREAT(FILENAME,(NSPAIR*(NVMAX+4)+319)/320,3,0,ISTAT,
C 4110 & TEMPV(NSPAIR*NVNLMAX+1))
C 4110 IF(ISTAT.EQ.0) GO TO 4120
C 4110 WRITE(06,3057)FILENAME,ISTAT
C 4110 IF(IMODE.EQ.1) GO TO 4111
C 4110 CALL PTIME(TIMEOUT)
C 4110 WRITE(FILEHO,3063)(TIMEOUT-TIMEIN)*3600.
C 4110 WRITE(06,*) ' NAMING OUTPUT HYPERPLANE FILE ',FILEHO
C 4110 GO TO 4112
C 4111 WRITE(06,*) ' NAME OUTPUT HYPERPLANE FILE ( ... TO OMIT) '
C 4111 READ(05,*)FILEHO
C 4111 IF(IMODE.EQ.0)WRITE(06,*)FILEHO
C 4112 IF(FILEHO.EQ.FILECR) GO TO 6000
C 4112 GO TO 4110
C 4120 CALL ATTACH(07,FILENAME,3,0,ISTAT,1)
C 4120 IF(ISTAT.NE.ISTATOK.AND.ISTAT.NE.ISTAT2)WRITE(06,240)ISTAT
C *****END OF OBSOLETE SECTION
C *****START OF NEW SECTION
4110 INQUIRE(FILE=FILEHO,EXIST=EXIS)
  IF (.NOT.EXIS) THEN
    OPEN(07,FILE=FILEHO,STATUS='NEW',IUSTAT=ISTAT,
      & ACCESS='SEQUENTIAL',FORM='FORMATTED')
    IF (ISTAT.NE.0) THEN

```

```

WRITE(06,*) ' ERROR OPENING OUTPUT HYPERPLANE FILE. ISTAT='
WRITE(06,*) ISTAT
      & STOP
      END IF
      IF (EXIS) THEN
        IF (IMODE.EQ.1) THEN
          WRITE(06,*) ' OUTPUT HYPERPLANE FILE', FILEHO,
            & ' ALREADY EXISTS.'
          WRITE(06,*) ' ENTER 1 TO OVERWRITE OR 2 TO ENTER NEW NAME.'
          READ(05,*) IOVERW
          IF (IOVERW.NE.1 .AND. IOVERW.NE.2) THEN
            WRITE(06,*) ' MUST ENTER 1 OR 2. PLEASE TRY AGAIN.'
            GO TO 4115
          END IF
          IF (IOVERW.EQ.1) THEN
            OPEN(07, FILE=FILEHO, STATUS='OLD', IOSTAT=ISTAT,
              & ACCESS='SEQUENTIAL', FORM='FORMATTED')
            IF (ISTAT.NE.0) THEN
              WRITE(06,*) ' ERROR OPENING OUTPUT HYPERPLANE FILE.',
                & ISTAT
              STOP
            END IF
            OPEN(07, FILE=FILEHO, STATUS='OLD', IOSTAT=ISTAT,
              & ACCESS='SEQUENTIAL', FORM='FORMATTED')
            IF (ISTAT.NE.0) THEN
              WRITE(06,*) ' ERROR OPENING OUTPUT HYPERPLANE FILE.',
                & ISTAT
              STOP
            END IF
            IF (IOVERW.EQ.2) THEN
              WRITE(06,*) ' ENTER NAME OF OUTPUT HYPERPLANE FILE',
                & ' (.... IF NONE)'
              READ(05,*) FILEHO
              IF (FILEHO.EQ.FILECR) GO TO 6000
              GO TO 4110
            END IF
          END IF
          IF (IMODE.EQ.0) THEN
            CALL PTIME(TIMEOUT)
            WRITE(FILINI,3063)INT((TIMEOUT-TIMEIN)*3600.*1000.)
            WRITE(06,*) ' NAMING OUTPUT HYPERPLANE FILE', FILEHO
            OPEN(07, FILE=FILEHO, STATUS='NEW', IOSTAT=ISTAT,
              & ACCESS='SEQUENTIAL', FORM='FORMATTED')
            IF (ISTAT.NE.0) THEN
              WRITE(06,*) ' ERROR OPENING OUTPUT HYPERPLANE FILE.',
                & ISTAT
              STOP
            END IF
          END IF
        END IF
      END IF
      C*****END OF NEW SECTION
      CALL FMEDIA(07,6)
      C
      K=0
      DO 4125 I=1, ISETNAM-1
        DO 4125 J=1, ISETNAM
          K=K+1
        DO 4121 JJ=1, NY
          4121 TEMPM(NSPAIR+1, JJ)=TEMPM(K, JJ)
          DO 4122 JJ=1, INMAX
            4122 TEMPM(K, JJ+4)=0
            ITEMPM(K, 1)=ISETID(1)
            ITEMPM(K, 2)=ISETID(2)
            TEMPM(K, 3)=P(1, K)-TEMPV(K)

```

```

TEMPM(K,4)=P(NVPJ-1,K)-TEMPV(K)
DO 4123 JJ=1,NV
4123 TEMPM(K,4+IN(JJ))=TEMPM(NSPAIR+1,JJ)
4125 CONTINUE
C
KK=NSPAR
IF(FILEHI.EQ.FILELC) GO TO 4170
4130 DO 4135 K=1,INMAX+4
4135 BUFFER(K)=0.
READ(04,*,END=4170)((BUFFER(L),L=1,2),(BUFFER(L),L=3
&,INMAX+4)
K=0
DO 4140 I=1, ISETNAM-1
DO 4140 J=I+1, ISETNAM
K=K+1
IF(ISETID(1).EQ.IBUFFER(1).AND.ISETID(J).EQ.IBUFFER(2))
& GO TO 4145
IF(ISETID(1).EQ.IBUFFER(2).AND.ISETID(J).EQ.IBUFFER(1))
& GO TO 4145
4140 CONTINUE
GO TO 4160
4145 DO 4146 JJ=3, INMAX+4
IF(TEMPM(K,JJ).NE.BUFFER(JJ)) GO TO 4148
4146 CONTINUE
GO TO 4130
4148 WRITE(06,4147)I,J
4147 FORMAT(' CHOOSE TO SAVE OLD,NEW PLANES (0,N) FOR SETS',I4,
&',' ,I4)
IF(IMODE.EQ.1) GO TO 4149
CHCF=CHO
WRITE(06,*) ' CHOOSING OLD PLANES WITHOUT COMPARISON WITH NEW'
GO TO 4155
4149 READ(05,*)CHCF
IF(CHCF.EQ.CHN) GO TO 4130
4155 TEMPM(K,1)=IBUFFER(1)
ITEMPM(K,2)=IBUFFER(2)
DO 4150 JJ=3, INMAX+4
4150 TEMPM(K,JJ)=BUFFER(JJ)
GO TO 4130
4160 KK=KK+1
ITEMPM(KK,1)=IBUFFER(1)
ITEMPM(KK,2)=IBUFFER(2)
DO 4165 J=3, I'MAX+4
4165 TEMPM(KK,J)=BUFFER(J)
GO TO 4130
4170 DO 4180 K=1,KK
WRITE(07,*) (ITEMPM(K,J),J=1,2),(TEMPM(K,J),J=3,INMAX+4)
4180 CONTINUE
4200 IF(FILEHI.NE.FILELC) REWIND 04
REWIND 07
WRITE(06,*) ' HYPERPLANE OUTPUT FILE WRITTEN'
CLOSE(04,STATUS='KEEP')
CLOSE(07,STATUS='KEEP')
C LINKS ENDS =====
6000 IF(IOUTPC.EQ.0) GO TO 9000
ICLNAME=0
MAINCL(1)=0
AXLENG(1)=0.
CG(1,1)=0.
C LINKC BEGINS =====

```



```

C CALC SOIST MATRIX OF STEPPING STONE DISTANCE BETWEEN PTS. NP*NP
C CALC IDIST MATRIX OF ASSOCIATED NUMBER OF STEPS BETWEEN PTS. NP*NP
C COMPUTE IFRONT MATRIX OF FRONTIER PT GOING FROM I(ROW) TO J(COL)
C SOISTMX IS THE MAX SS DIST BETWEEN AND TWO POINTS.
DO 6020 L=1, NP
DO 6010 K=1, NP
IFRONTIR(K)=L
IFRONTIC(K)=K
6010 IFRONTIR(L)=0
IFRONTIC(L)=0
WRITE(14,ERR=6210)((IFRONTIR(K),K=1, NP)
WRITE(15,ERR=6210)((IFRONTIC(K),K=1, NP)
6020 CONTINUE
REWIND 14
REWIND 15
CALL DATIM( DATE1, TIMES1)
CLSECDLI=120.
CLSECDLI=120.
ICHANGE1=0
IFSDN=22
IFSDO=12
IFIDN=23
IFIDU=13
IFIFRN=24
IFIFRO=14
IFIFCN=25
IFIFCO=15
6030 ICHANGE2=0
DO 6200 I=1, NP
CALL DATIM( DATE1, TIMES1)
IF(1.EQ.1) GO TO 6060
IF(1.EQ.2) GO TO 6050
DO 6040 L=1, I-1
READ(1FSDO,END=6210,ERR=6210)(BUFFER(K),K=1, NP)
WRITE(1FSDN,ERR=6210)(BUFFER(K),K=1, NP)
READ(1FIDO,END=6210,ERR=6210)(IBUFFER(K),K=1, NP)
WRITE(1FIDN,ERR=6210)(IBUFFER(K),K=1, NP)
READ(1FIFRO,END=6210,ERR=6210)(IBUFFER(K),K=1, NP)
WRITE(1FIFRN,ERR=6210)(IBUFFER(K),K=1, NP)
READ(1FIFCO,END=6210,ERR=6210)(IBUFFER(K),K=1, NP)
WRITE(1FIFCN,ERR=6210)(IBUFFER(K),K=1, NP)
CONTINUE
6040
6050 READ(1FSDO,END=6210,ERR=6210)(SDISTI(K),K=1, NP)
WRITE(1FSDN,ERR=6210)(SDISTI(K),K=1, NP)
READ(1FIDO,END=6210,ERR=6210)(IBUFFER(K),K=1, NP)
WRITE(1FIDN,ERR=6210)(IDISTI(K),K=1, NP)
READ(1FIFRO,END=6210,ERR=6210)(IBUFFER(K),K=1, NP)
WRITE(1FIFRN,ERR=6210)(IFRONTIR(K),K=1, NP)
READ(1FIFCO,END=6210,ERR=6210)(IBUFFER(K),K=1, NP)
WRITE(1FIFCN,ERR=6210)(IFRONTIC(K),K=1, NP)
6060 READ(1FSDO,END=6210,ERR=6210)(SDISTI(K),K=1, NP)
WRITE(1FSDN,ERR=6210)(SDISTI(K),K=1, NP)
READ(1FIDO,END=6210,ERR=6210)(IDISTI(K),K=1, NP)
WRITE(1FIDN,ERR=6210)(IDISTI(K),K=1, NP)
READ(1FIFRO,END=6210,ERR=6210)(IFRONTIR(K),K=1, NP)
WRITE(1FIFRN,ERR=6210)(IFRONTIR(K),K=1, NP)
READ(1FIFCO,END=6210,ERR=6210)(IFRONTIC(K),K=1, NP)
WRITE(1FIFCN,ERR=6210)(IFRONTIC(K),K=1, NP)
IF(1.EQ.NP) GO TO 6190
DO 6180 J=1+1, NP

```

```

READ(I,FSOO,END=6210,ERR=6210)((SDISTJ(K),K=1,NP)
READ(I,FI00,END=6210,ERR=6210)((IDISTJ(K),K=1,NP)
READ(I,FI00,END=6210,ERR=6210)((IFRONTJR(K),K=1,NP)
READ(I,FI00,END=6210,ERR=6210)((IFRONTJC(K),K=1,NP)
I1=I
J1=J
IJ=IVECTOR(I1,J1)
IDISTJ=IDISTJ(IJ)
TSDISTJ=SDISTJ(IJ)
ITEMP1=IFRONTIR(IJ)
ITEMP2=IFRONTJR(IJ)
ITEMP=0
DO 6160 K=1,NP
IF(K.EQ.1.OR.K.EQ.J) GO TO 6160
K1=K
TEMP1=AMAX1(SDISTJ(K),SDISTJ(K))
TEMPO=TSDISTJ
TSDISTJ=AMIN1(TEMPO,TEMP1)
IF(TEMPO-TEMP1) 6160,6070,6120
6070 IF(IDISTJ-IDISTJ(K)-IDISTJ(K)) 6160,6080,6120
6080 IFRONTJ=IFRONTIR(IJ)
IFRONTJ=IFRONTJR(IJ)
IFRONTIK=IFRONTIR(K)
IFRONTI=IFRONTIC(K)
IFRONTJK=IFRONTJR(K)
IFRONTKJ=IFRONTJC(K)
IF(SDISTJ(K),SDISTJ(K)) 6090,6100,6110
6090 IF(SDISTJ(IFRONTKJ),LT.SDISTJ(IFRONTJ))IFRONTIR(J)=
&IFRONTJC(K)
IFRONTJC(I)=IFRONTIR(J)
IF(SDISTJ(IFRONTJK),LT.SDISTJ(IFRONTJ))IFRONTJR(I)=
&IFRONTJR(K)
IFRONTIC(J)=IFRONTJR(I)
GO TO 6160
6100 IF(SDISTJ(IFRONTIK),LT.SDISTJ(IFRONTJ))IFRONTIR(J)=
&IFRONTIR(K)
IFRONTJC(I)=IFRONTIR(J)
IF(SDISTJ(IFRONTKJ),LT.SDISTJ(IFRONTJ))IFRONTIR(J)=IFRONTIKJ
IFRONTJC(I)=IFRONTIR(J)
IF(SDISTJ(IFRONTJK),LT.SDISTJ(IFRONTJ))IFRONTJR(I)=
&IFRONTJR(K)
IFRONTIC(J)=IFRONTJR(I)
IF(SDISTJ(IFRONTKI),LT.SDISTJ(IFRONTJ))IFRONTJR(I)=
&IFRONTIC(K)
IFRONTIC(J)=IFRONTJR(I)
GO TO 6160
6110 IF(SDISTJ(IFRONTIK),LT.SDISTJ(IFRONTJ))IFRONTIR(J)=IFRONTIK
IFRONTJC(I)=IFRONTIR(J)
IF(SDISTJ(IFRONTKI),LT.SDISTJ(IFRONTJ))IFRONTJR(I)=IFRONTKI
IFRONTIC(J)=IFRONTJR(I)
GO TO 6160
6120 ITEMP=K
IDISTJ=IDISTJ(K)+IDISTJ(K)
IF(SDISTJ(K)-SDISTJ(K)) 6130,6140,6150
6130 IFRONTIR(J)=IFRONTJC(K)
IFRONTJC(I)=IFRONTIR(J)
IFRONTJR(I)=IFRONTJR(K)
IFRONTIC(J)=IFRONTJR(I)
GO TO 6160
6140 IFRONTIR(J)=IFRONTIR(K)

```

```

IFRONTJC(I)=IFRONTIR(J)
IFRONTJR(I)=IFRONTJR(K)
IFRONTIC(J)=IFRONTJR(I)
GO TO 6100
6150 IFRONTIR(J)=IFRONTIR(K)
IFRONTJC(I)=IFRONTIR(J)
IFRONTJR(I)=IFRONTIC(K)
IFRONTIC(J)=IFRONTJR(I)
6160 CONTINUE
IF(IITEMP1.NE.IFRONTIR(J).OR.ITEMP2.NE.IFRONTJR(I))ICHANGE2=
&ICHANGE2+1
IF(IITEMP.EQ.0) GO TO 6170
ICHANGE2=ICHANGE2+1
6170 SDISTI(J)=TSDISTIJ
SDISTJ(I)=TSDISTIJ
SDISTMX=AMAX1(SDISTMX,TSDISTIJ)
IDIST1(J)=IDISTIJ
IDISTJ(I)=IDISTIJ
WRITE(IFSDN,ERR=6210)(SDISTJ(K),K=1,NP)
WRITE(IFIDN,ERR=6210)(IDISTJ(K),K=1,NP)
WRITE(IFIFRN,ERR=6210)(IFRONTJR(K),K=1,NP)
WRITE(IFIFCN,ERR=6210)(IFRONTJC(K),K=1,NP)
6180 CONTINUE
6190 REWIND IFSDN
REWIND IFSDO
REWIND IFIDN
REWIND IFIDO
REWIND IFIFRN
REWIND IFIFRO
REWIND IFIFCN
REWIND IFIFCO
ITEMPO=IFSDN
IFSDN=IFSDC
IFSDO=ITEMPO
ITEMPO=IFIDN
IFIDN=IFIDU
IFIDO=ITEMPO
ITEMPO=IFIFRN
IFIFRN=IFIFRO
IFIFRO=ITEMPO
ITEMPO=IFIFCN
IFIFCN=IFIFCO
IFIFCO=ITEMPO
CALL DATIM(DATE1,TIMEC1)
CLSEC1=(TIMEC1-TIMEI)*3600.
IF(CLSEC1.LT.CLSECDLI) GO TO 6200
WRITE(06,6195)ICHANGE1,I,ICHANGE2
6195 FORMAT(' ON CLUSTER ITERATION',I3,' TO POINT',I5,' CHANGED',
&I3,' SS DISTANCES')
6200 CONTINUE
IF(ICHANGE2.EQ.0) GO TO 6220
ICHANGE1=ICHANGE1+1
CALL DATIM(DATE1,TIMEC)
CLSEC=(TIMEC-TIMEI)*3600.
IF(CLSEC.LT.CLSECDLI) GO TO 6030
WRITE(06,6205)ICHANGE1,ICHANGE2
6205 FORMAT(' ON CLUSTER ITERATION',I3,' CHANGED',I8,' STEPPING',
&STONE DISTANCES')
WRITE(06,6206)(TIMEC-TIMEI)/60.
6206 FORMAT(' USING',F6.1,' MINUTES')

```

```

TIMES=TIMEC
TIMEST=TIMEC
WRITE(06,*) ENTER 0,1 TO CONCLUDE,CONTINUE CLUSTER ITERATION
READ(05,*)ICON
IF(ICON.EQ.0) GO TO 6220
IF(IMODE.EQ.0)WRITE(06,*)ICON
IF(ICON.EQ.0) GO TO 6220
GO TO 6030

C
6210 WRITE(06,*) STOP. IO ERROR IN HANDLING SDIST,IDIST,IFRONT
    & FILES
    STOP 5
6220 IF(IFSD0.EQ.12) GO TO 6230
    CLOSE(22,STATUS='DELETE')
    CLOSE(23,STATUS='DELETE')
    CLOSE(24,STATUS='DELETE')
    CLOSE(25,STATUS='DELETE')
    GO TO 6240
6230 CLOSE(12,STATUS='DELETE')
    CLOSE(13,STATUS='DELETE')
    CLOSE(14,STATUS='DELETE')
    CLOSE(15,STATUS='DELETE')
6240 IF(IDIGIT(IOUTPC,IOMAXC-1).EQ.0) GO TO 6280
    WRITE(06,*) MATRIX OF STEPPING STONE DISTANCES BETWEEN POINTS
    DO 6250 I=1,NP
    IF(NP.GT.10) WRITE(06,*) ROW',I
    READ(1FSDN,END=6260,ERR=6260)(BUFFER(K),K=1,NP)
    WRITE(06,70)(BUFFER(K),K=1,NP)
6250 CONTINUE
    GO TO 6270
6260 WRITE(06,*) NOTE. ERROR IN READING SDIST FILE.
6270 REWIND IFSDN
6280 IF(IDIGIT(IOUTPC,IOMAXC-2).EQ.0) GO TO 6320
    WRITE(06,*)
    WRITE(06,*) MATRIX OF COUNT OF STEPS BETWEEN POINTS
    DO 6290 I=1,NP
    IF(NP.GT.15) WRITE(06,*) ROW',I
    READ(1FIDN,END=6300,ERR=6300)(IBUFFER(K),K=1,NP)
    WRITE(06,60)(IBUFFER(K),K=1,NP)
6290 CONTINUE
    GO TO 6310
6300 WRITE(06,*) NOTE. ERROR IN READING IDIST FILE.
6310 REWIND IFIDN
6320 IF(IDIGIT(IOUTPC,IOMAXC-3).EQ.0) GO TO 6370
    WRITE(06,*)
    WRITE(06,*) MATRIX OF FRONTIER POINTS GOING FROM POINT I
    & (ROW) TO POINT J (COLUMN)
    DO 6340 J=1,NP
    READ(1FIFRN,END=6360,ERR=6360)(IBUFFER(K),K=1,NP)
    IPRINT=1
6330 WRITE(06,60)(IBUFFER(K),K=IPRINT,MIN0(NP,IPRINT+14))
    IPRINT=IPRINT+15
    IF(IPRINT.LE.NP) GO TO 6330
6340 CONTINUE
    REWIND IFIFRN
6350 CONTINUE
    GO TO 6370
6360 WRITE(06,*) NOTE. ERROR IN READING IFRONT FILE.
    REWIND IFIFRN

```

```

C COMPUTE ICLUSTER MATRIX OF POINTS (J) IN CLUSTER I (ROW)
C CALC IPINCL VECTOR. IPINCL(I) IS CLUSTER TO WHICH POINT I BELONGS
C IN ICLUSTER & IPINCL INTERIOR POINTS ARE -, OTHERS ARE FRONTIER
C POINTS. NPINCL(I) IS COUNT OF POINTS IN ITH CLUSTER. ICLNAME IS
C COUNT OF CLUSTERS. SDISTC IS MATRIX OF SS DISTANCES BETWEEN
C CLUSTERS WITH DIAGONALS THE MAX SS DIST WITHIN CLUSTERS.
6370 DO 6380 J=1,NP
      IPINCL(J)=0
      JBOND(J)=0
      DO 6380 I=1,ICLMAX
        ICLUSTER(I,J)=0
      DO 6390 I=1,NPMAX
        JBOND(IBOND(J))=1
        ICLNAME=1
        II=1
        IFLOCR=1
        C MARK PT II WITH -1 INDICATING IT IS PROCESSED OR ABOUT TO BE
        6400 ICLUSTER(ICLN,II)=-1
        IF (IBOND(II).EQ.0) GO TO 6410
        IF (ICLUSTER(ICLN,IBOND(II)).EQ.0) ICLUSTER(ICLN,
          &IBOND(II))=1
        6410 IF (JBOND(II).EQ.0) GO TO 6420
        IF (ICLUSTER(ICLN,JBOND(II)).EQ.0) ICLUSTER(ICLN,
          &JBOND(II))=1
        6420 DO 6450 L=1,IABS(IFLOCR-II)
          IF (II-IFLOCR) 6430,6450,6440
        6430 BACKSPACE IFIFRN
        GO TO 6450
        6440 READ(IFIFRN,END=6460,ERR=6460)(IBUFFER(K),K=1,NP)
        6450 CONTINUE
        READ(IFIFRN,END=6460,ERR=6460)(IBUFFER(K),K=1,NP)
        IFLOCR=II+1
        GO TO 6470
        6460 WRITE(06,*) ' ERROR IN READING IFRONT FILE.'
        STOP /
        6470 DO 6490 J=1,NP
          IF (J.EQ.II) GO TO 6490
          C PUT FRONTIER POINTS BELONGING TO II INTO ITS CLUSTER
          JJ=IBUFFER(J)
          IF (ICLUSTER(ICLN,JJ).NE.0) GO TO 6490
          ICLUSTER(ICLN,JJ)=1
          IF (IBOND(JJ).EQ.0) GO TO 6480
          IF (ICLUSTER(ICLN,IBOND(JJ)).EQ.0) ICLUSTER(ICLN,
            &IBOND(JJ))=1
        6480 IF (JBOND(JJ).EQ.0) GO TO 6490
          IF (ICLUSTER(ICLN,JBOND(JJ)).EQ.0) ICLUSTER(ICLN,
            &JBOND(JJ))=1
        6490 CONTINUE
        DO 6500 I=1,NP
          IF (ICLUSTER(ICLN,II).NE.1) GO TO 6500
        C PICK FIRST UNPROCESSED POINT IN CLUSTER, CALL IT II, RETURN TO
        C PROCESS IT
        II=1
        GO TO 6400
        6500 CONTINUE
        DO 6550 II=1,NP
          IF (ICLUSTER(ICLN,II).NE.0) GO TO 6550
          DO 6530 L=1,IABS(IFLOCR-II)
            IF (II-IFLOCR) 6510,6530,6520

```

```

6510 BACKSPACE IFIFRN
GO TO 6530
6520 READ(IFIFRN,END=6460,ERR=6460)(IBUFFER(K),K=1,NP)
6530 CONTINUE
READ(IFIFRN,END=6460,ERR=6460)(IBUFFER(K),K=1,NP)
IFLOC=11+1
DO 6540 J=1,NP
IF(11.EQ.J) GO TO 6540
C FIND PT HAVING A FRONTIER PT IN CLUSTER. CALL IT 11, PUT IN
C CLUSTER, ETC
IF(ICLUSTER(ICLNAME,IBUFFER(J)).NE.0) GO TO 6400
6540 CONTINUE
6550 CONTINUE
REWIND IFIFRN
IFLOC=1
C
C FIND SUBCLUSTERS OF LAST CLUSTER. POINTS IN SUBCLUSTER HAVE SAME
C FRONTIER IN GOING TO ANY OTHER SUBCLUSTER. LOOK FOR SUBMATRICES
C OF IFRONT HAVING THE MOST IDENTICAL ENTRIES FOR I&J BELONGING TO
C CLUSTER. (LOOP TO 6710) THEN FIND THE COLUMNS WITH DUPLICATE
C ENTRIES. (6660 TO 6710)
H=1
IFLOC=1
6560 KMIN=NP
KMAX=1
DO 6570 I=1,NP
IF(ICLUSTER(ICLNAME,I).EQ.0) GO TO 6570
KMIN=MIN0(KMIN,I)
KMAX=MAX0(KMAX,I)
6570 CONTINUE
KOUNT0=0
DO 6650 I=KMIN,KMAX
IF(ICLUSTER(ICLNAME,I))6580,6650,6580
6580 DO 6640 J=KMIN,KMAX
IF(ICLUSTER(ICLNAME,J))6590,6640,6590
6590 KOUNT=0
IFLOC=J
CALL FILELOC(IFIFRN,IFLOC,IFLOCN)
IFLOC=IFLOCN+1
READ(IFIFRN,END=6720,ERR=6720)(IBUFFER(K),K=1,NP)
DO 6620 K=J,KMAX
IF(ICLUSTER(ICLNAME,K))6600,6620,6600
6600 IF(IBUFFER(I)-IBUFFER(K))6610,6610,6620
6610 KOUNT=KOUNT+1
6620 CONTINUE
IF(KOUNT.LE.KOUNT0) GO TO 6640
KOUNT0=KOUNT
IO=1
JO=J
6640 CONTINUE
6650 CONTINUE
IF(KOUNT0.GT.1) GO TO 6660
H=H+1
GO TO 6730
IF(JO.EQ.KMAX.OR.IO.EQ.KMAX) GO TO 6730
IF(ICLNAME+H.LE.ICLMAX) GO TO 6670
WRITE(06,*) ' ALLOWABLE NUMBER OF CLUSTERS PREVENTS SUBDIVISION
& OF CLUSTER',ICLNAME
H=H+1
GO TO 6730

```

```

6670 ICLUSTER(ICLNAME+H,J0)=ICLJSTER(ICLNAME,J0)
IFLOCN=J0
CALL FILELOC(IFIFCN,IFLOC,IFLOCN)
IFLOC=IFLOCN+1
READ(IFIFCN,END=6720,ERR=6720)((IBUFFER(K),K=1,NP)
IFRONT0=IBUFFER(10)
J1=J0
DO 6690 J=J0+1,KMAX
IF(ICLUSTER(ICLNAME,J).EQ.0) GO TO 6690
IFLOCN=J
CALL FILELOC(IFIFCN,IFLOC,IFLOCN)
IFLOC=IFLOCN+1
READ(IFIFCN,END=6720,ERR=6720)((IBUFFER1(K),K=1,NP)
DO 6680 K=10,KMAX
IF(ICLUSTER(ICLNAME,K).EQ.0) GO TO 6680
IF(IBUFFER(K).NE.IFRONT0) GO TO 6680
IF(IBUFFER1(K).NE.IFRONT0) GO TO 6690
6680 CONTINUE
ICLUSTER(ICLNAME+H,J)=ICLUSTER(ICLNAME,J)
J1=J
6690 CONTINUE
IF(11.NE.J0) GO TO 6700
ICLUSTER(ICLNAME+H,J0)=0.
H=H-1
GO TO 6730
6700 DO 6710 J=J0,KMAX
IF(ICLUSTER(ICLNAME,J).EQ.ICLUSTER(ICLNAME+H,J))
&ICLUSTER(ICLNAME,J)=0.
6710 CONTINUE
H=H+1
GO TO 6560
6720 WRITE(06,*) ' ERROR WHILE READING IFRONTC FILE '
STOP 8
6730 REWIND IFIFCN
C
ICLNAME=ICLNAME+H
C COMBINE CLUSTERS THAT ARE BONDED
DO 6820 J1=1,NP
IF(IBOND(J1).EQ.0) GO TO 6820
J2=IBOND(J1)
DO 6740 K=1,ICLNAME
IF(ICLUSTER(K,J1).EQ.0) GO TO 6740
I1=K
GO TO 6750
6740 CONTINUE
GO TO 6820
6750 DO 6760 K=1,ICLNAME
IF(ICLUSTER(K,J2).EQ.0) GO TO 6760
I2=K
GO TO 6770
6760 CONTINUE
GO TO 6820
6770 IF(I1.EQ.I2) GO TO 6820
DO 6780 J=1,NP
6780 ICLUSTER(I1,J)=ICLUSTER(I1,J)+ICLUSTER(I2,J)
IF(I2.EQ.ICLNAME) GO TO 6800
DO 6790 I=12,ICLNAME-1
DO 6790 J=1,NP
6790 ICLUSTER(I,J)=ICLUSTER(I+1,J)
6800 DO 6810 J=1,NP

```

```

6810 ICLUSTER(ICLNAME,J)=0
ICLNAME=ICLNAME+1
6820 CONTINUE
DO 6850 J=1,NP
C
DO 6830 I=1,ICLNAME
IF(ICLUSTER(I,J).NE.0) GO TO 6850
6830 CONTINUE
IF(ICLNAME.LT.ICLMAX) GO TO 6840
WRITE(06,*) ' MORE CLUSTERS THAN ALLOWED BY DIMENSION: ',ICLMAX
GO TO 6890
6840 ICLNAME=ICLNAME+1
I1=J
GO TO 6400
6850 CONTINUE
C
C COMBINE IF ONLY ONE POINT PER CLUSTER
6860 IF(ICLNAME.NE.NP) GO TO 6890
DO 6880 J=1,NP
ICLUSTER(1,J)=1
DO 6870 I=2,ICLNAME
ICLUSTER(I,J)=0
6880 CONTINUE
ICLNAME=1
C
C
6890 DO 6900 J=1,NP
DO 6900 I=1,ICLNAME
IF(ICLUSTER(I,J).EQ.0) GO TO 6900
IF(IPINCL(J).NE.0) WRITE(06,*) ' POINT ',J, ' BELONGS TO CLUSTER
& ',IABS(IPINCL(J)), ' & ',I
IPINCL(I)=-I
6900 CONTINUE
DO 6910 I=1,NP
READ(IFIFRN,END=6920,ERR=6920)(IBUFFER(J),J=1,NP)
DO 6910 J=1,NP
IF(I.EQ.J) GO TO 6910
IF(IABS(IPINCL(I)).EQ.IABS(IPINCL(J)).AND.IABS(IPINCL(I)).EQ.
& IABS(IPINCL(IBUFFER(J)))) GO TO 6910
IPINCL(IBUFFER(J))=IABS(IPINCL(IBUFFER(J)))
6910 CONTINUE
REWIND IFIFRN
GO TO 6930
6920 WRITE(06,*) ' ERROR WHILE READING IFIFRN FILE '
STOP 9
6930 DO 6940 I=1,NP
IF(IPINCL(I).LT.0) GO TO 6940
IF(IPINCL(I).EQ.0) WRITE(06,*) ' ERROR. POINT ',I, ' UNASSIGNED
& TO CLUSTER
ICLUSTER(IPINCL(I),I)=IABS(ICLUSTER(IPINCL(I),I))
6940 CONTINUE
IF(DIGIT(OUTPUTC,IOMXC-4).EQ.0) GO TO 7010
WRITE(06,*)
WRITE(06,*) ' NUMBER OF THE CLUSTER TO WHICH POINT BELONGS
& (-MEANS INTERIOR POINT)
DO 6950 I=1,(NP+9)/10
IPRINT=1+(I-1)*10
WRITE(06,BU) (IPINCL(J),J=IPRINT,MINU(NP,IPRINT+9))
6950 CONTINUE
IF(ISETNAM.LE.1) GO TO 7010

```


NO 4236 899

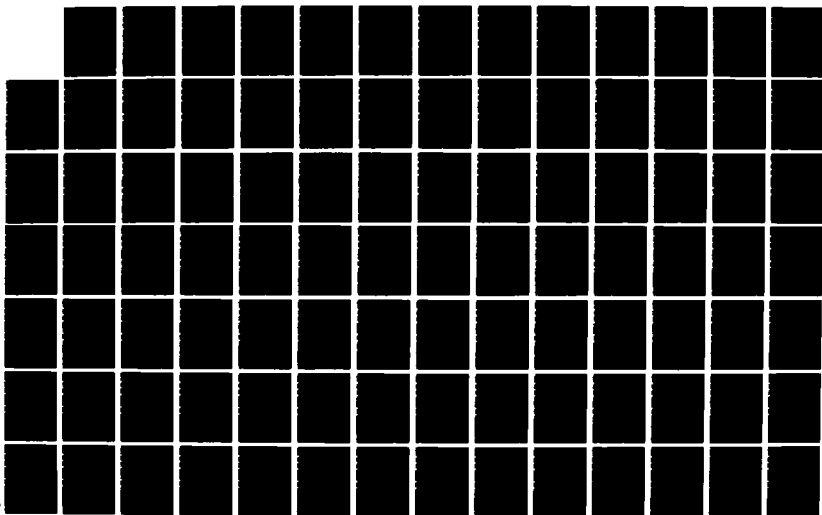
EMPIRICAL FLUTTER PREDICTION METHOD(U) GE AIRCRAFT
ENGINES CINCINNATI OH ADVANCED TECHNOLOGY OPERATION
J K CASEY 85 MAR 88 R87AEG AFMAL-TR-87-2887

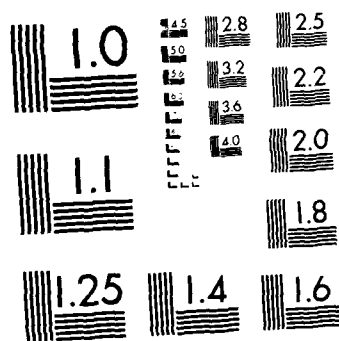
374

UNCLASSIFIED

F/G 20/4

NL





```

WRITE(06,*) COUNT OF SET MEMBERS IN EACH CLUSTER
WRITE(06,*) I=1, ISETNAM
DO 7003 I=1, ISETNAM
  WRITE(06,*) SET ID = ISETID(I)
  DO 6960 J=1, ICLNAME
    ITEMPV(J)=0
    DO 6980 J=1, ICLNAME
      DO 6970 K=1, NP
        IF (ABS(IPINSET(K)).EQ.I.AND.ICLUSTER(J,K).NE.0) ITEMPV(J)=
          &ITEMPV(J)+1
      CONTINUE
    CONTINUE
    IPRINT=1+(J-1)*10
    WRITE(06,80) (ITEMPV(L), L=IPRINT, MINO(ICLNAME, ICLNAME+9))
  CONTINUE
  DO 6990 J=1, (ICLNAME+9)/10
    IPRINT=1+(J-1)*10
    WRITE(06,80) (ITEMPV(L), L=IPRINT, MINO(ICLNAME, ICLNAME+9))
  CONTINUE
  DO 7010 I=1, ICLNAME
    NPINCL(I)=0
    DO 7020 J=1, NP
      IF (ICLUSTER(I,J).EQ.0) GO TO 7020
      NPINCL(I)=NPINCL(I)+1
    CONTINUE
  CONTINUE
  DO 7040 K=1, ICLNAME
    DO 7030 KK=1, ICLNAME
      SDISTC(K, KK)=SDISTMX
      DO 7040 SDISTC(K, K)=0
      DO 7050 I=1, NP
        K=IABS(IPINCL(I))
        READ(1FSDN, END=6350, ERR=6350) (BUFFER(L), L=1, NP)
        DO 7050 II=1, NP
          KK=IABS(IPINCL(II))
          IF (K.EQ.KK) SDISTC(K, K)=AMAX1(SDISTC(K, K), BUFFER(II))
          IF (K.NE.KK) SDISTC(K, KK)=AMIN1(SDISTC(K, KK), SDISTC(KK, K),
            &BUFFER(II))
        DO 7050 SDISTC(KK, K)=SDISTC(K, KK)
        IF (IFSDN.EQ.12) REWIND 12
        IF (IFSDN.EQ.22) REWIND 22
        IF (IDIGIT(IOUTPC, IOMAXC-5).EQ.0) GO TO 7070
        WRITE(06,*) STEPPING STONE DISTANCES FOR CLUSTERS (ON DIAG=
          &WITHIN, OFF DIAG=BETWEEN)
        DO 7060 K=1, ICLNAME
          WRITE(06,70) (SDISTC(K, J), J=1, ICLNAME)
        CONTINUE
      CONTINUE
    CONTINUE
  CONTINUE
  C FOR POINTS IN EACH CLUSTER SUBTRACT CG AND PUT IN BOTTLE
  C FORM XINF INFORMATION MATRIX. THIS IS XINF WITH POINTS GROUPED BY
  C CLUSTER, WITH CG OF CLUSTER SUBTRACTED, ROTATED ALONG AXES OF
  C BOTTLE. AXES HAS IN EACH ROW A SEMI AXIS OF LENGTH 1, UP TO NVMAX
  C COLUMNS. THERE ARE NVMAX ROWS RESERVED PER CLUSTER, UP TO ICLMAX
  C COLUMNS. AXLENG HAS FORMAT OF A COLUMN OF AXES, HAS LENGTH OF
  C CORR ROW OF AXES. NAINCL(K) IS THE NUMBER OF AXES IN CLUSTER K.
  C THIS CODING IS USED FOR SETS AS WELL AS FOR CLUSTERS.
  DO 7070 C=SETCL-CHCLS
    NSETCL=ICLNAME
    IOUTPSC=IOUTPC
    IOUTSC=0

```

```

DO 7072 I=1, ICLNAME
7072 NPIN(I)=NPINCL(I)
C LINKC ENDS =====
7075 II=0
      III=1
      IV=1
DO 7080 I=1, NAXES
7080 AXLENG(I)=0.
DO 7090 I=1, NAXES
DO 7090 J=1, NVMAX
7090 AXES(I,J)=0.
      ITEMPV(I)=1
DO 7100 I=2, NSETCL
7100 ITEMPV(I)=ITEMPV(I-1)+
      &NPIN(I-1)
      I=0
7110 READ(10,END=7130)(BUFFER(L),L=1,NV)
C TO SCREEN DATA FROM FILE, ACTIVATE STATEMENTS AS IN S7115
C7115 IF(BUFFER(IACCEP).NE.VACCEPT) GO TO 7110
      I=I+1
      IF(CHSETCL.EQ.CHCL(I)) *=[ABS(IPINCL(I))
      IF(CHSETCL.EQ.CHSET) *=[ABS(IPINSET(I))
DO 7120 J=1, NV
7120 XINF(ITEMPV(K),J)=BUFFER(J)
      ITEMPV(K)=ITEMPV(K)+1
      IF(I.LT.NP) GO TO 7110
7130 REWIND 10
C-----
DO 7340 K=1, NSETCL
      II=II+NPIN(K)
      TEMP=NPIN(K)
DO 7160 J=1, NV
      CG(K,J)=0.
DO 7140 I=III, II
7140 CG(K,J)=CG(K,J)+XINF(I,J)
      CG(K,J)=CG(K,J)/TEMP
DO 7150 I=III, II
7150 XINF(I,J)=XINF(I,J)-CG(K,J)
7160 CONTINUE
      CALL BOTTLE (XINF(III,1),AXES(IV,1),AXLENG(IV),NV,NPIN(K))
      NAINCL(K)=0
DO 7170 I=1, NV
      IF(AXLENG(IV+I-1).NE.0)NAINCL(K)=I
7170 CONTINUE
DO 7190 I=1, NAINCL(K)-1
DO 7190 II=I+1, NAINCL(K)
      TEMP=0.
DO 7180 J=1, NV
7180 TEMP=TEMP+AXES(IV+I-1,J)*AXES(IV+II-1,J)
      IF(ABS(TEMP).LE.10.E-5) GO TO 7190
      WRITE(06,7185)I,II,TEMP*AXLENG(IV+I-1)*AXLENG(IV+II-1)
7185 FORMAT(' AXES',2I4,' NOT ORTHOGONAL.DOT PRODUCT IS ',E12.6)
7190 CONTINUE
      IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-6).EQ.0) GO TO 7230
      WRITE(06,*)
      IF(NPIN(K).GT.1) GO TO 7200
      WRITE(06,*) ' SEMIAXES OF BOTTLE FOR ',CHSETCL,K,' NULL SINCE 1
      & POINT.
      GO TO 7240
7200 WRITE(06,7210)CHSETCL,K

```

```

7210 FORMAT(' SEMIAXES OF BOTTLE CONTAINING ',A7,I6)
DO 7220 I=1,NAINCL(K)
WRITE(06,70)(AXES(I+I-1,J)*AXLENG(I+I-1),J=1,NV)
7220 CONTINUE
WRITE(06,*) ' LENGTH OF THE SEMIAXES'
WRITE(06,70)(AXLENG(I+I-1),I=1,NAINCL(K))
C
7230 IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-7).EQ.0) GO TO 7240
WRITE(06,*)
WRITE(06,*) ' PROJECTION OF ',CHSETCL,K,' ON ITS NORMALIZED AXES'
WRITE(06,*) ' AXIS PROJECTION/AXIS LENGTH'
CALL PLOTSCAT(XINF(III,1),AXLENG(IV),NV,NPIN(K))
7240 IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-8).EQ.0) GO TO 7330
WRITE(06,*)
IF(NAINCL(K).GT.0) GO TO 7250
WRITE(06,*) ' DATA TRANSFORMS BETWEEN BOTTLE & ORIGINAL AXES NULL'
GO TO 7330
7250 WRITE(06,7260)CHSETCL,K
7260 FORMAT(' DATA TRANSFORM TO ',A7,I3,' AXES (V) FROM'
& ORIGINAL AXES (U)')
DO 7290 I=1,NAINCL(K)
TEMP=0.
DO 7270 J=1,NV
7270 TEMP=TEMP-AXES(IV+I-1,J)*CG(K,J)
DO 7280 J=1,NV
IF(AXES(IV+I-1,J).GE.0.) CHS(J)=CHP
IF(AXES(IV+I-1,J).LT.0.) CHS(J)=CHM
7280 CONTINUE
WRITE(06,100)CHV,I,TEMP,(CHS(J),ABS(AXES(IV+I-1,J)),CHU,J,
&J=1,MINO(3,NV))
IF(NV.LE.3) GO TO 7290
WRITE(06,110)((CHS(J),ABS(AXES(IV+I-1,J)),CHU,J),J=4,NV)
7290 CONTINUE
WRITE(06,7300)CHSETCL,K
7300 FORMAT(' DATA TRANSFORM FROM ',A7,I3,' AXES (V) TO ORIGINAL'
& AXES (U)')
DO 7320 J=1,NV
DO 7310 I=1,NAINCL(K)
IF(AXES(IV+I-1,J).GE.0.) CHS(I)=CHP
IF(AXES(IV+I-1,J).LT.0.) CHS(I)=CHM
7310 CONTINUE
WRITE(06,100)CHU,J,CG(K,J),(CHS(I),ABS(AXES(IV+I-1,J))),
&CHV,I,I=1,MINO(3,NAINCL(K))
IF(NAINCL(K).LE.3) GO TO 7320
WRITE(06,110)((CHS(I),ABS(AXES(IV+I-1,J)),CHV,I),I=4
&,NAINCL(K))
7320 CONTINUE
7330 I1=I1+NPIN(K)
IV=IV+NV
7340 CONTINUE
IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-9).EQ.0) GO TO 7360
WRITE(06,*)
WRITE(06,*) ' CENTER OF GRAVITY OF EACH ',CHSETCL
DO 7350 K=1,NSETCL
WRITE(06,70)(CG(K,J),J=1,NV)
7350 CONTINUE
C
7360 IF(CHSETCL.EQ.CHSET) GO TO 7465
IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-9).NE.0) WRITE(06,*) '
DO 7370 K=1,NSETCL

```

```

ITEMPV(K)-0
7370 ITEMPPV(K+NSETCL)=0
NFILE=1
CALL ATFILE(FILIN,NFILE)
I=0
7380 READ(U,90,END=7410)(BUFFER(L),L=1,INMAX)
C TO EXCLUDE DATA POINTS, ACTIVATE STATEMENTS LIKE S7385
C7385 IF(BUFFER(IACCEP).NE.VACCEP) GO TO 7380
I=I+1
KK=IABS(IIPINCL(I))
DO 7400 K=1,NSETCL
TEMP=0.
DO 7390 J=1,NV
7390 TEMP=TEMP+((BUFFER(IN(J))-STYP(J))/SVAR(J)-CG(K,J))**2
TEMP=SQRT(TEMP)
IF(KK.NE.K.AND.TEMP.LT.SDISTC(K,KK))ITEMPV(K+NSETCL)=KK
IF(KK.EQ.K.AND.TEMP.LE.SDISTC(K,K))ITEMPV(K)=0
7400 CONTINUE
IF(I.LT.NP) GO TO 7380
7410 NFILE=NFILE+1
CALL ATFILE(FILIN,NFILE)
IF(NFILE.NE.U) GO TO 7380
DO 7460 K=1,NSETCL
ITEMP1=ITEMPV(K)
ITEMP2=ITEMPV(K+NSETCL)
IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-9).EQ.0) GO TO 7460
IF(ITEMP1.EQ.0.AND.ITEMP2.EQ.0) WRITE(06,7420)CHSETCL,K,
& CHSETCL
IF(ITEMP1.NE.0.AND.ITEMP2.NE.0) WRITE(06,7430)CHSETCL,K,
& CHSETCL,CHSETCL,ITEMP2
IF(ITEMP1.EQ.0.AND.ITEMP2.NE.0) WRITE(06,7440)CHSETCL,K,
& CHSETCL,CHSETCL,ITEMP2
IF(ITEMP1.NE.0.AND.ITEMP2.EQ.0) WRITE(06,7450)CHSETCL,K,
& CHSETCL
7420 FORMAT(A7,I4,' CG INSIDE ITS ',A7)
7430 FORMAT(A7,I4,' CG OUTSIDE ITS ',A7,' TOWARD ',A7,I4)
7440 FORMAT(A7,I4,' CG INSIDE ITS ',A7,' TOWARD ',A7,I4)
7450 FORMAT(A7,I4,' CG OUTSIDE ITS ',A7,' AWAY FROM OTHERS')
7460 CONTINUE
C-----
7465 III=1
DO 7610 K=1,NSETCL
KK=(K-1)*NV
IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-10).EQ.0) GO TO 7610
WRITE(06,*),
& CHSETCL
IF(NPIN(K).GT.1) GO TO 7480
WRITE(06,7470)CHSETCL,K
C CORRELATION MATRIX FOR POINTS IN 'A7,16,' NULL
& SINCE 1 POINT )
III=III+NPIN(K)
GO TO 7610
7480 TEMP=NPIN(K)
ITEMP=0
DO 7490 I=1,NV
IF(AXLENG(KK+I).NE.U.)ITEMP=I
7490 CONTINUE
IF(ITEMP.GT.0) GO TO 7510
WRITE(06,*), ' STOP. AXES HAVE LENGTH ZERO.'
7500 STOP 407
7510 DO 7530 I=1,ITEMP

```

```

DO 7530 J=1,ITEMP
  VAR(I,J)=0.
DO 7520 I=1,111-NPIN(K)-1
  7520 VAR(I,J)=VAR(I,J)+XINF(11,I)*XINF(11,J)
  VAR(I,J)=VAR(I,J)/TEMP
  VAR(J,I)=VAR(I,J)
7530 CONTINUE
  III=111-NPIN(K)
  IF(IDIGIT(IOUTPSC,IOMAXC-IOUTSC-10).EQ.0) GO TO 7560
  WRITE(06,7540)CHSETCL,K
  7540 FORMAT(' CORRELATION MATRIX FOR NEW COMPONENTS OF POINTS IN
    &A7.16)
DO 7550 I=1,ITEMP
  WRITE(06,70)(VAR(I,J)/SQRT(VAR(I,I)*VAR(J,J)),J=1,ITEMP)
7550 CONTINUE
C SKIP DETERMINANT CALC DUE TO 1) DISCREP IN DIM OF TEMP IN DETERM
C & DTRMIN 2) TEMP FOR DTRMIN IS CALC AS NP*NV, BUT DIM
C IS DIFFERENT 3) TERPM HERE OVERWRITES TERPMV AT
C $ 975 4) DETERMINANT NOT CHECKED OUT ANYHOW.
7560 GO TO 7595
DO 7570 I=1,ITEMP
DO 7570 J=1,ITEMP
  7570 TEMP(I,J)=VAR(I,J)
  CALL DETERM (TEMP,ITEMP,CVOLUME)
DO 7580 I=1,NPIN(K)
DO 7580 J=1,ITEMP
  7580 TEMP(I,J)=XINF(111-NPIN(K)+1-1,J)
  CALL DTRMIN(TEMP,NPIN(K),ITEMP,DETER,KOUNT)
7595 CONTINUE
  WRITE(06,*)' CVOLUME',SQRT(DETER*(TEMP**ITEMP))
  WRITE(06,*)K,NPIN(K),ITEMP,DETER,KOUNT,III
  CVOLUME=SQRT(CVOLUME*(TEMP**ITEMP))
  BVOLUME=1.
DO 7590 I=1,ITEMP
  BVOLUME=BVOLUME*AXLENG(KK+1)*2.
  WRITE(06,7600)CHSETCL,CVOLUME,BVOLUME,CVOLUME/BVOLUME
  7600 FORMAT(A7,' VOLUME/BOTTLE VOLUME=',E12.4,' / ',E12.4,' =',
    &E12.4)
  7610 CONTINUE
C-----
  IF(CHSETCL.EQ.CHCLS) GO TO 7615
  CHSETCL=CHCLS
  IOUTPSC=IOUTPL
  IOUTSC=0
  GO TO 970
C=====
  7615 IF(IDIGIT(IOUTPL,IOMAXC-11).EQ.0) GO TO 9000
  III=1
DO 7720 K=1,ICLNAME
  IF(NPINCL(K).GT.1) GO TO 7630
  WRITE(06,*)K
  WRITE(06,7620)K
  7620 FORMAT(' QUADRATIC EQUATION FIT TO CLUSTER',I3, ' NULL SINCE
    & ONLY ONE POINT')
  GO TO 7720
  7630 DO 7640 I=1,NAINCL(K)
DO 7640 J=1,NAINCL(K)
  7640 IJ=NAINCL(K)*(I-1)-(I*(I-1))/2+J
  IJ(IJ)=I
  JIJ(IJ)=J

```

```

7640 CONTINUE
      NANINCL(K)=(NAINCL(K)*(NAINCL(K)+1))/2
      DO 7660 I=1,NAINCL(K)
      DO 7660 J=1,NAINCL(K)
      IJ=NAINCL(K)*(I-1)-(I*(I-1))/2+J
      TEMPV(IJ)=0
      DO 7650 H=1,NAINCL(K)
      IF(Q(H).EQ.0.) GO TO 7650
      TEMPV(IJ)=TEMPV(IJ)+XINF(III-1+I,H)*XINF(III-1+J,H)/(Q(H)**2)
7650 CONTINUE
      IF(I.NE.J) TEMPV(IJ)=2.*TEMPV(IJ)
7660 CONTINUE
      DO 7670 I=1,NANINCL(K)
      CHS(I)=CHP
      IF(TEMPV(I).LT.0.)CHS(I)=CHM
7670 CONTINUE
      WRITE(06,*)
      WRITE(06,7680)K
7680 FORMAT(' QUADRATIC EQUATION FIT TO CLUSTER',I3,' USING CLUSTER
      & AXES (V)')
      WRITE(06,7690)(CHS(I),ABS(TEMPV(I)),IJ(I),J(I),I=1,
      &MINO(3),NANINCL(K))
7690 FORMAT(' 1=',A2,E10.3,'*V(',I2,',I2,')V(',I2,',I2,')',A2,E10.3,'*V(
      & ,I2,',I2,')',A2,E10.3,'*V(',I2,',I2,')V(',I2,',I2,')')
      IF(NANINCL(K).LE.3) GO TO 7710
      WRITE(06,7700)(CHS(I),ABS(TEMPV(I)),IJ(I),J(I),I=1,
      &NANINCL(K))
7700 FORMAT(' ,A2,E10.3,'*V(',I2,',I2,')V(',I2,',I2,')',
      & ,A2,E10.3,'*V(',I2,',I2,')V(',I2,',I2,')')
7710 III=III+NPINCL(K)
7720 CONTINUE
C=====
C TERMINATING RUN PROCEDURE
5000 WRITE(06,*)
      CALL PTIME(TIMEOUT)
      CALL DATIM(DATE1,TIME1)
      WRITE(06,9010)((TIMEOUT-TIMEIN)*3600.,TIME1
9010 FORMAT(' TIME USED:',F8.2,' SECONDS, COMPLETED AT',
      &F6.2,' HOURS')
      WRITE(06,*)' ENTER 0,N NOTTO,TO TRANSFORM DATA PER CLUSTER
      & N AXES, N<0 PER ALL DATA'
      READ(05,*)ICASE2
      IF(IMODE.EQ.0)WRITE(06,*)ICASE2
      IF(ICASE2.EQ.0) GO TO 9140
9020 WRITE(06,*)' ENTER MAX NUMBER OF AXES TO BE USED'
      READ(05,*)ITEMP1
      IF(IMODE.EQ.0)WRITE(06,*)ITEMP1
      IF(ITEMP1.LT.1.AND.IMODE.EQ.1) GO TO 9020
      IF(ITEMP1.LT.1.AND.IMODE.EQ.0) STOP 16
      NFILE=1
      CALL ATFILE(FILIN,NFILE)
      I=0
      NTEMP=0
9030 READ(01,90,END=9050)(BUFFER(L),L=1,INMAX)
      NTEMP=NTEMP+1
      IF(NSETOUT.LE.0) GO TO 9035
      DO 9034 K=1,NSETOUT
      IF(IBUFFER(ISETVAR).EQ.ISETOUT(K)) GO TO 9030
9034 CONTINUE

```



```

9035 IF(NPTOUT.LE.0) GO TO 9039
DO 9037 K=1,NPTOUT
IF(NPTMP.EQ.IPTOUT(K)) GO TO 9030
9037 CONTINUE
9039 I=I+1
DO 9040 J=1,MAXJ
9040 XINF(I,J)=(BUFFER(IN(J))-STVP(J))/
&SVAR(J)
9050 NFILE=NFILE+1
CALL ATFILE(FILIN,NFILE)
IF(NFILE.NE.0) GO TO 9030
IF(I.LT.NP) GO TO 9030
IF(ICASE2.GT.0) GO TO 9090
ITEMP=0
TEMP1=AXLTOT(I)
DO 9060 I=1,NV
IF(AXLTOT(I).EQ.0.) GO TO 9060
ITEMP=I
TEMP1=AMIN1(TEMP1,AXLTOT(I))
9060 CONTINUE
ITEMP=MIND(ITEMP,ITEMP1)
DO 9080 I=1,NP
DO 9080 J=1,ITEMP
TEMP=0.
DO 9070 K=1,NV
9070 TEMP=TEMP+(XINF(I,K)-CGTOT(K))*AXTOT(J,K)
XINF(I,J)=TEMP*TEMP1/AXLTOT(J)
9080 CONTINUE
NV=ITEMP
WRITE(06,*)' '
WRITE(06,140)
WRITE(06,*)' '
WRITE(06,*)' '
GO TO 450
9090 ITEMP=(ICASE2-1)*NV
ITEMP=MIND(ITEMP,ITEMP1)
TEMP1=AXLENG(1+ITEMP)
DO 9100 J=1,NAINCL(ICASE2)
TEMP1=AMIN1(TEMP1,AXLENG(J+ITEMP))
9100 CONTINUE
ITEMP=(ICASE2-1)*NV
DO 9120 I=1,NP
DO 9120 J=1,NAINCL(ICASE2)
TEMP=0.
DO 9110 K=1,NV
9110 TEMP=TEMP+(XINF(I,K)-CG(ICASE2,K))*AXES(J+ITEMP,K)
XINF(I,J)=TEMP*TEMP1/AXLENG(J+ITEMP)
9120 CONTINUE
NV=NAINCL(ICASE2)
WRITE(06,140)
WRITE(06,*)' '
WRITE(06,9130)ICASE2
WRITE(06,*)' '
9130 FORMAT(' ',I4)
DATA TRANSFORMED PER AXES OF CLUSTER
GO TO 450
9140 IF(ICLNAME.LE.1) GO TO 9170
WRITE(06,*)' ENTER 0,1 IFNOT,IF ABOVE CLUSTER CG'S TO BE
& TREATED AS POINTS'
READ(05,*)ICASE1

```



```

TEMP1=0.
DO 60 I=1,NP1
TEMP2=0.
DO 50 J=1,NV
50 TEMP2=TEMP2+A(I,J)**2
TEMP2=ABS(TEMP2)
IF(TEMP1.GE.TEMP2) GO TO 60
TEMP1=TEMP2
ITEMP1=I
60 CONTINUE
TEMP1=TEMP1
TEMP1=TEMP1
C STORE IN X WITH LENGTH IN XLAMBDA
DO 70 I=1,NV
70 X(K,I)=A(ITEMP1,I)/TEMP1
XLAMBDA(K)=TEMP1
C MODIFY MATRIX A
ITEMP2=0
DO 120 I=1,NP1
IF(I.NE.ITEMP1) GO TO 90
DO 80 J=1,NV
80 A(I,J)=0.
ANU(I,K)=XLAMBDA(K)
GO TO 120
90 TEMP3=0.
DO 100 J=1,NV
100 TEMP3=TEMP3+X(K,J)*A(I,J)
ANU(I,K)=TEMP3
DO 110 J=1,NV
A(I,J)=A(I,J)-TEMP3*X(K,J)
IF(A(I,J).NE.0.) ITEMP2=I
110 CONTINUE
120 CONTINUE
IF(ITEMP2.NE.0.AND.K.LT.NV.AND.K.LT.NP1-1) GO TO 40
125 DO 145 I=1,NP1
DO 130 J=1,K
130 A(I,J)=ANU(I,J)
IF(K.GE.NVMAX) GO TO 145
DO 135 J=K+1,NVMAX
135 A(I,J)=0.
145 CONTINUE
150 RETURN
END
CDTERM=====
SUBROUTINE DETERM(A,N,DETER)
PARAMETER (NPMAX=450,NVMAX=14,ICLMAX=14)
PARAMETER (NAXES=NVMAX*ICLMAX)
PARAMETER (NVNLMAX=NVMAX*(NVMAX+1)/2)
COMMON LNPMAX,LNVMAX,LICLMAX,LNAXES,LLWK,WK
DIMENSION A(NVNLMAX,NVNLMAX)
IF(NVMAX.NE.LNVMAX) WRITE(06,*) ' IN DETERM NVMAX=',NVMAX,
& ' BUT IN MAIN NVMAX= ',LNVMAX
DO 50 K=1,N-1
DO 30 I=K+1,N
DO 30 J=K+1,N
30 A(I,J)=A(I,J)-A(K,J)*A(I,K)/A(K,K)
50 CONTINUE
DETER=1.
DO 60 I=1,N
60 DETER=DETER*A(I,I)
RETURN

```

```

END
COTRMIN-----
SUBROUTINE DTRMIN(A,N,M,DETER,KOUNT)
PARAMETER (NP=450,NVMAX=14,ICLMAX=14)
COMMON LNPMAX,LNVMAX,LICLMAX,LNAXES,LLWK,WK
DIMENSION A(NVMAX,NVMAX),ITEMP(NVMAX)
IF(NVMAX.NE.LNVMAX)WRITE(06,*) IN DTRMIN NVMAX=',NVMAX,
&' BUT IN MAIN NVMAX=',LNVMAX
DO 5 K=1,NVMAX
5 ITEMP(K)=0
KOUNT=0
DO 50 L=1,N-1
AMAX=0.
IMAX=0.
DO 10 K=1,N
IF(A(K,K).LE.AMAX.OR.ITEMP(K).NE.0) GO TO 10
AMAX=A(K,K)
IMAX=K
10 CONTINUE
IF(AMAX.EQ.0.) GO TO 60
ITEMP(IMAX)=1
KOUNT=KOUNT+1
DO 30 I=1,N
IF(I.EQ.IMAX) GO TO 30
TEMP=A(I,IMAX)/A(IMAX,IMAX)
DO 20 J=1,M
20 A(I,J)=A(I,J)-A(IMAX,J)*TEMP
A(I,IMAX)=0.
30 CONTINUE
IF(KOUNT.EQ.N.OR.KOUNT.EQ.M) GO TO 60
50 CONTINUE
60 I SWITCH=0
WRITE(06,*) ' 7596' ,K,AMAX,IMAX,KOUNT,ITEMP,
&((A(I,J), J=1,M),I=1,N)
DO 70 K=1,NVMAX
IF(ITEMP(K).EQ.0) GO TO 70
IF(I SWITCH.EQ.0) DETER=A(K,K)
IF(I SWITCH.NE.0) DETER=DETER*A(K,K)
I SWITCH=1
70 CONTINUE
RETURN
END
CINSECT-----
SUBROUTINE INSECT(X,ID,K,H,INV,INP,TEMP,ITEMP,IER)
C IN PLACE OF DUAL SOLUTION, IS LISTED THE NON 0 COMPONENTS OF
C PRIMAL SOLN, THEIR NUMBER, THEIR LOCATION IN PRIMAL SOLN VECTOR.
PARAMETER (NPMAX=450,NVMAX=14,ICLMAX=14)
PARAMETER (NAXES=NVMAX*ICLMAX)
COMMON LNPMAX,LNVMAX,LICLMAX,LNAXES,LLWK,WK
INTEGER H
DIMENSION X(NPMAX,NVMAX),ID(1),TEMP(1),ITEMP(1),IW(1)
IF(NVMAX.GE.INV+4) GO TO 5
WRITE(06,4)INV,NVMAX-4
4 FORMAT(' INSECT VARIABLES COUNT IS INV=',14,' BUT MAX IS
& NVMAX-4=',14)
IER=135
GO TO 300
5 IF(NPMAX.EQ.LNPMAX) GO TO 10
WRITE(06,*) ' IN INSECT NPMAX=',NPMAX,' BUT IN MAIN NPMAX=',
& LNPMAX

```

```

IER=135
GO TO 300
10 IF(NVMAX.EQ.LNVMAX) GO TO 15
WRITE(06,*) ' IN INSECT NVMAX= ', NVMAX, ' BUT IN MAIN NVMAX= ',
& LNVMAX
IER=135
GO TO 300
15 N=0
ITEMPO=2*INP*(INV+7)**2-6*INV
DO 100 J=1,INP
IF(ID(J).EQ.K) GO TO 20
IF(ID(J).EQ.H) GO TO 30
GO TO 100
20 TEMP1=1.
TEMP2=1.
TEMP3=0.
GO TO 40
30 TEMP1=-1.
TEMP2=0.
TEMP3=1.
40 N=N+1
DO 50 I=1,INV
50 TEMP(ITEMPO+I*(N-1)*NVMAX)-X(J,I)*TEMP1
TEMP(ITEMPO+INV+1*(N-1)*NVMAX)=TEMP2
TEMP(ITEMPO+INV+2*(N-1)*NVMAX)=TEMP3
TEMP(INV+4*N)-1
100 CONTINUE
DO 110 I=1,INV
110 TEMP(I)=0.
TEMP(INV+1)=1.
TEMP(INV+2)=1.
ITEMP1=ITEMPO+INP*INV
ITEMP2=(NVMAX*(NVMAX+1)/2)**2
IF(ITEMP1.LE.ITEMP2) GO TO 200
WRITE(06,*) ' IN INSECT NEED TEMP', ITEMPI, ' LONG; BUT IS ONLY ',
& ITEMPI, ' LONG'
IER=135
GO TO 300
200 CALL ZX3LP(TEMP(ITEMPO+1), NVMAX, TEMP, TEMP(INV+5), N, 0, INV+2,
& TEMP(N+INV+5), TEMP(N+INV+6), TEMP(2*N+2*INV+8), TEMP(2*N+
& 3*INV+12), TEMP(2*N*(INV+4)**2+5*(INV+4)), IER)
IF(IER.EQ.133) GO TO 300
N1=0
M1=0
DO 250 J=1,INP
TEMP1=0.
IF(ID(J).EQ.K) TEMP1=1.
IF(ID(J).EQ.H) TEMP1=-1.
IF(TEMP1.EQ.U) GO TO 250
N1=N1+1
IF(TEMP(N+INV+5*N1).EQ.U) GO TO 250
M1=M1+1
IF(M1.GT.INV+2) WRITE(06,*) ' ERROR. INSECT HAS MORE THAN ',
& INV+2, ' NON 0 COMPONENTS'
TEMP(2*N+2*INV+7*M1)=TEMP(N+INV+5*N1)*TEMP1
ITEMP(2*N+3*INV+10*M1)-J
250 CONTINUE
ITEMP(2*N+3*INV+10)=M1
300 RETURN
END

```

```

CEXP0-----
      FUNCTION EXP0(X)
      IF(X.LT.-60.) EXP0=0.
      IF(X.GE.-60.) EXP0=EXP(X)
      RETURN
      END
      FUNCTION ALOG0(X)
      IF(X)10,20,10
      10 ALOG0=ALOG(X)
      GO TO 30
      20 ALOG0=-60.
      GO TO 30
      30 RETURN
      END
CIDIGIT=====
      FUNCTION IDIGIT(I,X,I)
      C RETURNS IX WITH 0 FOR ALL BUT ITH DIGIT. IDIGIT(9876.3)=800.
      DOUBLE PRECISION IDIGIT,I*.1,1.12,ITEN
      ITEN=10
      I1=ITEN**I
      I2=ITEN**(I-1)
      IDIGIT=DMOD(I,X,I1)
      IF(I.GT.0) IDIGIT=IDIGIT-DMOD(I,X,I2)
      RETURN
      END
CLEAD0-----
      FUNCTION LEAD0(I)
      C EXAMPLE: LEAD0(850400)=7
      DO 10 J=1,11
      IF(MOD(I,10**(12-J)).NE.I)30 TO 20
      10 CONTINUE
      J=J+1
      20 LEAD0=13-J
      RETURN
      END
      FUNCTION LEAD0R(X)
      C EXAMPLE: LEAD0R(.05E20)=19,LEAD0R(1.)=1,LEAD0R(1.2340E-17)=-16
      DO 10 J=1,72
      IF(AMOD(X,10.**((36-J))-X)20,10,20
      10 CONTINUE
      J=J+1
      20 LEAD0R=J7-J
      RETURN
      END
      FUNCTION LAGO(I)
      C EXAMPLE: LAGO(850400)=3
      DO 10 J=1,11
      IF(MOD(I,10**(J-1)).NE.0)GO TO 20
      10 CONTINUE
      J=J+1
      20 LAGO=J-1
      RETURN
      END
CPLOTSCAT=====
      SUBROUTINE PLOTSCAT(A,X,LAMBDA,NV,NP1)
      PARAMETER (NPMAX=450,NVMAX=14,ICLMAX=14)
      COMMON LN1 MAX,LNVMAX,LI1CLMAX,LINAXES,LLWK,WK
      DIMENSION A(NPMAX,NVMAX),*LAMBDA(1),ITEMP(51),CH(10)
      CHARACTER *1CH,CHR(51)
      CH(1)=-1.

```

```

CH(2)=-2.
CH(3)=-3.
CH(4)=-4.
CH(5)=-5.
CH(6)=-6.
CH(7)=-7.
CH(8)=-8.
CH(9)=-9.
CH(10)=-10.
WRITE(06,*)
8 -8 -6 -4 -2 0 .2 .4 .6 .8 1.
IF(NPMAX.NE.LNPMAX) WRITE(06,*) IN PLOTSCAT NPMAX=,NPMAX,
B. BUT IN MAIN NPMAX=,LNPMAX
DO 60 J=1,NV
IF(XLAMBDA(J).EQ.0) GO TO 100
DO 20 I=1,51
20 ITEMP(I)=0
DO 30 I=1,NPI
I1=26.5+25.*A(I,J)/XLAMBECA(J)
I1=MAX0(I,I1)
I1=MIN0(51,I1)
ITEMP(I1)=ITEMP(I1)+1
30 CONTINUE
DO 35 I=1,51
35 CHR(I)=.
DO 40 I=1,51
ITEMPO=MIN0(10,ITEMP(I))
I1=1
IF(ITEMPO.EQ.0) GO TO 40
CHR(I1)=CH(ITEMPO)
40 CONTINUE
50 WRITE(06,50)J,(CHR(I),I=1,51)
50 FORMAT('6',51A1)
60 CONTINUE
100 RETURN
END
CIVECTR-----
FUNCTION IVECTR(I,J)
C COMPUTE VECTOR POSITION OF I,J ENTRY IN NPMAX*NPMAX SYMMETRIC
C MATRIX WHOSE UPPER TRIANGULAR PART IS NOT STORED.
C NOTE NPMAX MUST AGREE WITH NPMAX IN MAIN.
PARAMETER (NPMAX=450,N/MAX=14,ICLMAX=14)
I1=MAX0(1,J)
J1=MIN0(1,J)
IVECTR=(I1*(I1-1)/2)+J1
IF(I1.LE.NPMAX.AND.J1.LE.NPMAX) GO TO 10
WRITE(06,*) ERROR: IVECTR ARGUMENT EXCEEDS NPMAX.
X=SQRT(-1.)
10 RETURN
END
=====
SUBROUTINE FILELOC(IFILE,LOCOLD,LOCNEW)
C MOVES FILE FROM BEGIN OF RECORD LOCOLD TO BEGIN OF RECORD LOCNEW.
C NOTE THAT AFTER RECORD I HAS BEEN READ, LOCOLD WOULD BE I+1.
C STOPS IF READ ERROR
IF(LOCNEW.EQ.LOCOLD) GO TO 50
DO 30 K=1,IABS(LOCNEW-LOCOLD)
IF(LOCNEW-LOCOLD) 10,30,20
10 BACKSPACE IFILE
GO TO 30

```

```

20 READ(IFILE,ERR=40,END=40)(TEMP,L=1,10)
30 CONTINUE
GO TO 50
40 WRITE(06,*) ' ERROR IN FILELOC WHILE READING FILE ',IFILE
STOP
50 RETURN
END
CAAT=====
SUBROUTINE AAT(A,IA,N,M,W)
C A IS N*M MATRIX STORED IN AN ARRAY IA*MAX(N,M) WHICH WILL CONTAIN
C A*ATranspose UPON RETURN. W IS WORK VECTOR OF N WORDS.
DIMENSION A(1),W(1)
DO 60 I=1,N
DO 40 J=1,N
W(J)=0.
DO 30 K=1,M
30 W(J)=A(1+IA*(K-1))*A(J+IA*(K-1))+W(J)
40 CONTINUE
DO 50 J=1,N
50 A(1+IA*(J-1))=W(J)
60 CONTINUE
DO 70 I=2,N
DO 70 J=1,I-1
70 A(1+IA*(J-1))=A(J+IA*(I-1))
RETURN
END
COISCRIM=====
SUBROUTINE DISCRIM(X,ID,K,H,NV,NP,N,P,D,IER)
C DISCRIM SEEKS HYPERPLANE DESCRIBED BY VECTOR P, SCALAR D THAT
C SEPARATES POINTS (ROWS) IN X WHOSE ID IS K FROM THOSE WITH ID H.
C THUS FOR POINT Z, Z*P GE D OR LE D AS ID IS K OR H. N IS NUMBER
C OF POINTS IN THE TWO SETS. X IS AN NP*NV ARRAY WITHIN AN NPMAX
C *NVMAX ARRAY. WK IS WORK VECTOR OF LENGTH LB, A LITTLE LT
C (NV+2)*N+2*(N+4)**2 IF N GE NV. IER=0 IF SUCCESS;
C =33,34,70 IF WARNING FROM SV DECOMP OR LP ROUTINES; =129 IF SV
C DECOMP FAILS, =130-133 IF LP FAILS, =135 IF DIMENSIONS WRONG.
C
PARAMETER (NPMAX=450, NVMAX=14, ICLMAX=14)
PARAMETER (NAXES=NVMAX*ICLMAX)
COMMON LNPMAX, LNVMAX, LICLMAX, LMAXES, LLWK, WK
INTEGER H
DIMENSION ID(1), WK(1), P(1)
DIMENSION A(1), V(1), W(1), U(1), Q(1), TEMP(1)
DIMENSION B(1), C(1), PSOL(1), DSOL(1), RW(1), LW(1)
EQUIVALENCE (A(1),WK(1)), (V(1),WK(1)), (W(1),WK(1))
&, (U(1),WK(1)), (Q(1),WK(1)), (TEMP(1),WK(1)), (B(1),WK(1))
&, (C(1),WK(1)), (PSOL(1),WK(1)), (DSOL(1),WK(1)), (LW(1),WK(1)),
&, (RW(1),WK(1))
M=MAX0(N,NV+1)
L0=M*(NV+1)
L1=L0+1
L2=L1*(N+3)*N
L3=L2+3*
& MAX0(N,NV)
L4=L3+N
IF(LLWK.GE.L3*N*NV+NV+1) GO TO 5
WRITE(06,*) ' IN DISCRIM, WK REQUIRES MORE THAN L3*N*NV+NV+1 =
&, L3*N*NV+NV+1. WORDS BUT HAS AVAILABLE ONLY LLWK = ', LLWK
IER=135
GO TO 600

```



```

5 CONTINUE
IER=0
TOL=.001
IF(NPMAX.EQ.LNPMAX) GO TO 10
WRITE(06,*) ' IN DISCRIM NPMAX=',NPMAX, ' BUT IN MAIN NPMAX ='
&,LNPMAX
IER=135
GO TO 600
10 IF(NVMAX.EQ.LNPMAX) GO TO 15
WRITE(06,*) ' IN DISCRIM NVMAX=',NVMAX, ' BUT IN MAIN NVMAX='
&,LNPMAX
IER=135
GO TO 600
C PUT THE N POINTS OF X WITH ID K OR H INTO N*(NV+1) MATRIX A.
C IN COL NV+1 ENTER -1 IF POINT HAS ID=K, +1 IF POINT HAS ID=H.
15 L=0
DO 80 I=1,NP
READ(22)(WK(L4+J),J=1,NV)
IF(ID(1).EQ.K) GO TO 20
IF(ID(1).EQ.H) GO TO 30
GO TO 80
20 TEMP1=+1.
TEMP2=-1.
GO TO 40
30 TEMP1=-1.
TEMP2=+1.
40 L=L+1
IF(L.LE.N) GO TO 45
WRITE(06,*) ' DISCRIM COUNT EXCEEDS',N, ' POINTS WITH ID',K, ' OR',
& H
IER=136
GO TO 600
45 DO 50 J=1,NV
50 A(L+(J-1)*M)=TEMP1*WK(L4+J)
A(L+NV*M)=TEMP2
80 CONTINUE
C GET SINGULAR VALUE DECOMP OF N*(NV+1) MATRIX A AS U*Q*VTRANSPOSE
C WHERE U IS N*N (BUT IMBEDDED IN AN (N+3)*N ARRAY), Q IS N*(NV+1),
C V IS (NV+1)*2. ONLY THE NV+1 DIAGONAL WORDS OF Q ARE STORED. V
C STORED OVER A. U INITIALIZED TO IDENT. UTRANSPOSE STORED OVER U.
C TEMP IS 2(NV+1) WORDS LONG
DO 120 I=1,N
DO 110 J=1,N
110 U(L1+I+(J-1)*(N+3))=0.
120 U(L1+I+(I-1)*(N+3))=1.
CALL LSVDF(A,M,N,NV+1,U(L1+1),N+3,N,Q(L2+1),TEMP(L2+NV+2),
& IER)
IF(IER.EQ.33.OR.IER.EQ.34)WRITE(06,*) ' WARNING: LSVDF IN
& DISCRIM RETURNS IER=',IER
IF(IER.EQ.129) GO TO 600
QMAX=0
DO 130 I=1,NV+1
IF(ABS(Q(L2+I)).GT.QMAX)QMAX=ABS(Q(L2+I))
130 CONTINUE
DO 140 I=1,NV+1
IF(ABS(Q(L2+I)).LT.TOL*QMAX)Q(L2+I)=0.
140 CONTINUE
C REPLACE V BY W=VDQTUT, AN (NV+1)*N MATRIX. T=TRANSPOSE. D IS
C (NV+1)*2 GENERALIZED INVERSE OF QTQ.
DO 145 I=1,N*NV+(NV+1)

```

```

145 RW(L3+I+((I-1)/M)*(NV+1-M))=V(I)
DO 160 I=1,NV+1
DO 150 J=1,N
W(I+(J-1)*(NV+1))=0.
DO 150 L=1,NV+1
IF(Q(L2+L).EQ.0.) GO TO 150
W(I+(J-1)*(NV+1))=W(I+(J-1)*(NV+1))+RW(L3+I+(L-1)*(NV+1))*
&U(L1+L+(J-1)*(N+3))/Q(L2+L)
150 CONTINUE
160 CONTINUE
C SET UP LP PROBLEM: (D-1)UT*PSOL=B, SUM PSOL LE 1, PSOL GE 0.
C MAXIMIZE S=C*PSOL. IN CODE, INEQ CONSTRAINT AFFIXED AT BEGINNING
C OF U, AS ROW. SET UP LP PROBLEM: (D-1)UT*PSOL=B, MAXIMIZE S=C*PSOL
C PSOL GE 0. D IS DIAG MATRIX WITH II ENTRY 1,0 AS II ENTRY OF Q
C IS NE,EQ 0.
DO 165 J=1,N
165 U(L0+1+(J-1)*(N+3))=1.
ICONSTR=0
DO 180 I=1,N
IF(I.LE.NV+1.AND.Q(L2+I).NE.0.) GO TO 180
ICONSTR=ICONSTR+1
DO 170 J=1,N
170 U(L1+ICONSTR+(J-1)*(N+3))=U(L1+I+(J-1)*(N+3))
180 CONTINUE
B(L2+1)=1.
DO 175 I=2,ICONSTR+1
175 B(L2+I)=0.
DO 210 I=1,N
210 C(L3+I)=1.
L5=L4+MAX0(N,ICONSTR+1)
L6=L5+ICONSTR+3
L7=L6+2*ICONSTR+7
L8=L7+(ICONSTR+4)**2
IF(LLWK.GE.L8) GO TO 190
IER=135
WRITE(0G,*) ' IN DISCRIM, WK REQUIRES L8=',L8, ' WORDS BUT HAS
& AVAILABLE ONLY LLWK=',LLWK
GO TO 600
190 CALL ZX3LP(U(L0+1),N*3,B(L2+1),C(L3+1),N,1,ICONSTR,S,
&PSOL(L4+1),DSOL(L5+1),RW(L7+1),IW(L6+1),IER)
IF(IER.EQ.133) GO TO 600
C COMPUTE P AND D
DO 230 I=1,NV
P(I)=0.
DO 230 J=1,N
230 P(I)=P(I)+W(I+(J-1)*(NV+1))*PSOL(L4+J)
D=0.
DO 240 J=1,N
240 D=D+W(NV+1+(J-1)*(NV+1))*PSOL(L4+J)
600 REWIND 22
RETURN
END
CNOSMPX=====
SUBROUTINE NOSMPX (A,IA,M,N,IR,JCI,IRANK,X,IFEAS)
C SOLVES AX=B WITH X GE 0, USING NON OBJECTIVE SIMPLEX METHOD. X IS
C NON UNIQUE & IS RETURNED IN PLACE OF B. A IS M*N MATRIX, WITH ROW
C DIMENSION IA. ASSUMES THAT A CONTAINS L,U FACTORS AS PRODUCED BY
C LUFACTR BACKS, WITH OTHER ORIGINAL COLUMNS Q REPLACED BY UINV*
C LINV*Q. B HAS M COMPONENTS. X HAS N COMPONENTS, BUT ONLY M ARE
C NON 0. IN OUTPUT, COMPONENT X(IR(K)) MULTIPLIES COLUMN JC(K);

```

```

C OMITTED COMPONENTS =U. IFEAS=-NUMBER NEG COMPONENTS OF X; IFEAS=0
C IF LP PROBLEM IS 'FASIBLE.
  PARAMETER (NPMAX=450,NVMAX=14,ICLMAX=14)
  PARAMETER (NVMAXP1=NVMAX+1)
  DIMENSION A(1),IR(1),JC(1),IRI(1),JCI(1),X(1)
  &,THETAU(NVMAX),THETAL(NVMAXP1)
  COMMON LNPMAX,LNVMAX,LICLMAX,LNAXES,LLWK,WK
  IF(NVMAX.NE.LNVMAX) WRITE(6,*) ' IN NOSMPLEX NVMAX=',NVMAX,
  & BUT IN MAIN NVMAX=',LNVMAX
  XINFIN=1000.
  XINTES=1.E-6
  C REPLACE B BY UNV*LINV*B
  CALL BACKS1 (A,IA,M,N,IR,JC,IRI,JCI,IRANK,X)
  C REPLACE L,U IN A BY IDENTITY MATRIX
  DO 20 I=1,IRANK
    DO 20 J=1,IRANK
      20 A(IR(I)+(JC(J)-1)*IA)=0.
  DO 30 K=1,IRANK
    30 A(IR(K)+(JC(K)-1)*IA)=1.
  NKOUNT=0
  DO 50 I=1,IRANK
    IF(A(I).LT.0.)NKOUNT=NKOUNT+1
  50 CONTINUE
  IF(NKOUNT.EQ.0) GO TO 290

C
C SEARCH FOR REPLACEMENT COL FOR BASIS, WHICH WILL DECREASE NKOUNT
  ITHDIFR=1
  100 NKOUNT=NKOUNT
  IUR=0
  ILR=0
  KUR=0
  KLR=0
  DO 200 K=1,N
    IF(JCI(K).NE.0) GO TO 200
    C FOR COLUMN K OF A, DETERMINE UPPER,LOWER BOUNDS ON THETA
    KOUNTU=0
    KOUNTL=1
    THETAL(1)=0.
    IU=0
    IL=0
    KU=0
    KL=0
    DO 130 I=1,IRANK
      IF(ABS(A(IR(I)+(K-1)*IA)).LE.XINTES) GO TO 130
      IF(A(IR(I)+(K-1)*IA)) 120,130,110
    110 KOUNTU=KOUNTU+1
      THETAU(KOUNTU)=X(IR(I))/A(IR(I)+(K-1)*IA)
      IF(KOUNTU.EQ.1) GO TO 115
      IF(THUMIN.LE.THETAU(KOUNTU)) GO TO 130
    115 THUMIN=THETAU(KOUNTU)
      IU=1
      KU=K
    GO TO 130
  120 KOUNTL=KOUNTL+1
      THETAL(KOUNTL)=0.
      IF(ABS(X(IR(I))),LE.ABS(A(IR(I)+(K-1)*IA))*XINTES) GO TO 125
      THETAL(KOUNTL)=X(IR(I))/A(IR(I)+(K-1)*IA)
    125 IF(KOUNTL.EQ.2) GO TO 127
      IF(THLMAX.GE.THETAL(KOUNTL)) GO TO 130
    127 THLMAX=THETAL(KOUNTL)

```

```

IL=I
KL=K
130 CONTINUE
IF(KOUNTU.NE.0) THDIF=THUMIN-THLMAX
IF(KOUNTU.EQ.0) THDIF=XINFIN
C DETERMINE FOR EACH COLUMN IN BASIS, THE NUMBER KOUNTC OF NEGATIVE
C COMPONENTS IN X IF THAT COL IS REPLACED IN BASIS BY COL K OF A
DO 195 I=1,IRANK
NKCOUNTC=0
IF(A(IR(I))*(K-1)*IA)) 150,140,150
140 IF(X(IR(I)).GE.0.) GO TO 195
NKCOUNTC=NKCOUNTC+1
GO TO 195
150 TEMP=0.
IF(ABS(X(IR(I))).LE.ABS(A(IR(I))*(K-1)*IA))*XINTES) GO TO 155
TEMP=X(IR(I))/A(IR(I))*(K-1)*IA
155 IF(KOUNTL.EQ.0) GO TO 170
DO 160 KI=1,KOUNTL
IF(TEMP.GE.THETAL(KI)) GO TO 160
NKCOUNTC=NKCOUNTC+1
160 CONTINUE
170 IF(KOUNTU.EQ.0) GO TO 190
DO 180 KI=1,KOUNTU
IF(TEMP.LE.THETAU(KI)) GO TO 180
NKCOUNTC=NKCOUNTC+1
180 CONTINUE
190 IF(I.EQ.IU.AND.KU.EQ.K.AND.NKCOUNTC.GT.NKOUNT) IU=0
IF(I.EQ.IL.AND.KL.EQ.K.AND.NKCOUNTC.GT.NKOUNT) IL=0
IF(NKOUNTL.LE.NKCOUNTC) GO TO 195
NKOUNTR=NKOUNTC
IREM=I
KREM=K
195 CONTINUE
C IF NO REPLACEMENT WILL REDUCE NKOUNT, TRY TO INCREASE THDIF.
IF(NKOUNT.EQ.0) GO TO 210
IF(NKOUNT.NE.NKOUNT) GO TO 200
IF(IU.EQ.0.AND.IL.EQ.0) GO TO 200
IF(THDIFR.EQ.1) GO TO 199
IF(KOUNTU.EQ.0) GO TO 199
IF(THDIF.LE.THDIFFR) GO TO 200
199 THDIFFR=THDIF
THDIFFR=0
IUR=IU
ILR=IL
KUR=KU
KLR=KL
200 CONTINUE
C
C REPLACEMENT OF COL IR(IREM) IN BASIS BY COL KREM, WILL REDUCE
C NUMBER OF NEGATIVE COMPONENTS IN X FROM NKOUNT TO NKOUNTR
IF(NKOUNTR.NKOUNT) 210,205,209
205 IF(THDIFFR.EQ.1) GO TO 209
IF(KU.EQ.0.AND.KL.EQ.0) GO TO 209
IF(IUR.EQ.0.AND.ILR.EQ.0) GO TO 209
IREM=IUR
KREM=KUR
IF(IUR.NE.0) GO TO 215
IREM=ILR
KREM=KLR
GO TO 215

```

```

209 IFEAS=-NKOUNT
    GO TO 300
C UPDATE IR,JC,IRI,JCI,A,X,NKOUNT
210 NKOUNT=NKOUNT+1
    ITHDI=1
215 THETA=0.
    IF (ABS(X(IR(IREM)))) LE ABS(A(IR(IREM)+(KREM-1)*IA))*XINTES)
        & GO TO 220
    THETA=X(IR(IREM))/A(IR(IREM)+(KREM-1)*IA)
220 JJ=JC(IREM)
    JC(IREM)=KREM
    JCI(JJ)=0
    JCI(KREM)=IREM
    DO 240 I=1,IRANK
        TEMP=A(IR(I)+(KREM-1)*IA)
        IF (ABS(TEMP) LE XINTES) GO TO 240
        X(IR(I))=(X(IR(I))/TEMP-THETA)*TEMP
240 CONTINUE
        X(IR(IREM))=THETA
    DO 260 J=1,N
        IF (J EQ KREM) GO TO 260
    DO 250 I=1,IRANK
        A(IR(I)+(J-1)*IA)=A(IR(I)+(J-1)*IA)-A(IR(IREM)+(J-1)*IA)*
            & A(IR(I)+(KREM-1)*IA)/A(IR(IREM)+(KREM-1)*IA)
250 CONTINUE
        A(IR(IREM)+(J-1)*IA)=A(IR(IREM)+(J-1)*IA)/A(IR(IREM)+
            & (KREM-1)*IA)
260 CONTINUE
    DO 270 I=1,IRANK
        A(IR(I)+(KREM-1)*IA)=0.
        A(IR(IREM)+(KREM-1)*IA)=1.
C TERMINATION PROCEDURE
    IF (NKOUNT GT 0) GO TO 100
290 IFEAS=0
300 RETURN
END
CLUFACR-----
SUBROUTINE LUFACR (A,IA,M,N,IR,JC,IRI,JCI,IRANK)
C TRIANGULAR FACTORS L,U ARE GOTTEN FOR SQUARE SUBMATRIX OF THE M*N
C MATRIX A, WITH USE OF PIVOTING. THE KTH CHOSEN ROW, COL IN L*U-I
C IS IN LOCATION IR(K),JC(K) OF A. THE DIAGONAL OF L IS I AND NOT
C STORED. THE RANK OF A IS IRANK. REQUIRES M LE N. ANY COLUMN Q OF A
C NOT USED IN L,U HAS BEEN PROCESSED THE SAME AS THE LAST COL OF U,
C HAS BECOME LINV*Q. IA IS THE ROW DIMENSION OF A. IRI,JCI ARE THE
C INVERSES OF IR,JC IN SENSE IRI(ROW K)=ORDER IN WHICH ROW K
C INTRODUCED INTO BASIS; IS 0 IF NOT USED. DITTO FOR JC(COL K).
C THUS IRI(TIME K)=LOCATION H; IRI(LOCATION H)=TIME K
    DIMENSION A(1),IR(1),JC(1),IRI(1),JCI(1)
    IRANK=0
    DO 10 I=1,M
        IR(I)=0
        IRI(I)=0
        JC(I)=0
    DO 20 J=1,N
        JCI(J)=0
    DO 100 K=1,M
        ELMAX=0.
    DO 50 I=1,M
        IF (IRI(I) NE 0) GO TO 50

```

```

DO 40 J=1,N
IF (JCI(J).NE.0) GO TO 40
TEMP=A(I+(J-1)*IA)
IF (ABS(ELMAX).GE.ABS(TEMP)) GO TO 40
ELMAX=TEMP
II=I
JJ=J
40 CONTINUE
50 CONTINUE
IF (ELMAX.EQ.0) GO TO 300
IRI(II)=K
JCI(JJ)=K
IR(K)=II
JC(K)=JJ
IRANK=IRANK+1
DO 90 I=1,M
IF (IRI(I).NE.0) GO TO 90
A(I+(JJ-1)*IA)=A(I+(J-1)*IA)/ELMAX
DO 80 J=1,N
IF (JCI(J).NE.0) GO TO 80
A(I+(J-1)*IA)=A(I+(J-1)*IA)-A(II+(J-1)*IA)*A(I+(JJ-1)*IA)
80 CONTINUE
90 CONTINUE
100 CONTINUE
300 RETURN
END

C-----
SUBROUTINE BACKS(A,IA,M,N,IR,JC,IRI,JCI,IRANK)
C ASSUMES A HAS BEEN PROCESSED BY LUFACR. COLUMN Q OF A, NOT IN
C L,U WAS REPLACED BY LINV*Q IN LUFACR. HERE THIS IS REPLACED BY
C LINV*UINV*Q. THE L,U MATRICES ARE NOT ALTERED.
DIMENSION A(1),IR(1),JC(1),IRI(1),JCI(1)
DO 80 II=1,IRANK
III=IRANK+1-II
IF (III.EQ.IRANK) GO TO 60
DO 50 I=III+1,IRANK
TEMP=A(IR(III)+(JC(I)-1)*IA)
DO 40 J=1,N
IF (JCI(J).NE.0) GO TO 40
A(IR(III)+(J-1)*IA)=A(IR(III)+(J-1)*IA)-A(IR(I)+(J-1)*IA)*
& TEMP
40 CONTINUE
50 CONTINUE
60 DO 70 J=1,N
IF (JCI(J).NE.0) GO TO 70
A(IR(III)+(J-1)*IA)=A(IR(III)+(J-1)*IA)/A(IR(III)+(JC(III)-1)
& *IA)
70 CONTINUE
80 CONTINUE
RETURN
END

C-----
SUBROUTINE BACKS1(A,IA,M,N,IR,JC,IRI,JCI,IRANK,X)
C COMPUTES UINV*LINV*X AND STORES IN X. X HAS M COMPONENTS. A IS
C NOT CHANGED. ASSUMES L,U ARE IN A IN FORM PRODUCED BY LUFACR.
C ESSENTIALLY PERFORMS ON SEPARATE VECTOR X, WHAT BACKS DOES TO COL
C U OF A NOT IN L,U.
DIMENSION A(1),IR(1),JC(1),IRI(1),JCI(1),X(1)
C CALCULATE LINV*X=Y, ASSUMING DIAG PART OF L IS ALL 1'S
DO 80 II=1,IRANK

```

```

IF(I1.EQ.1) GO TO 80
DO 50 I=1,I1-1
50 X(IR(I1))=X(IR(I1))-X(IR(I1))*A(IR(I1))*(JC(I1)-1)*IA)
80 CONTINUE
C CALCULATE UINV*Y
DO 180 II=1,IRANK
II1=IRANK+1-II
IF(I11.EQ.IRANK) GO TO 160
DO 150 I=II1+1,IRANK
150 X(IR(I11))=X(IR(I11))-X(IR(I1))*A(IR(I11))*(JC(I1)-1)*IA)
160 X(IR(I11))=X(IR(I11))/A(IR(I11))*(JC(I11)-1)*IA)
180 CONTINUE
RETURN
END
=====
SUBROUTINE VERTEX (X,N,M,NMAX,MMAX,ID,IDMAX,TEMP,ITEMP)
C GIVEN N*M TABLE X OF N POINTS HAVING M COMPONENTS WITH ROW DIMEN-
C SION NMAX, AND VECTOR ID OF IDENTIFICATION NUMBERS 1 THRU IDMAX
C FOR THE N PTS OF X. FINDS THE 2M PTS WITH MAX OR MIN COMPONENT
C FOR EACH SET OF PTS WITH SAME ID. 1ST IDMAX ROWS OF TEMP HAVE PT
C NUMBERS. 1ST M WORDS ARE FOR MAX. 2ND M FOR MIN. 4*MMAX WORDS PER
C ROW. NEXT IDMAX ROWS FOR CORRESPONDING COMPONENT VALUES.
C TEMP/ITEMP IS AN ARRAY OF SIZE (2*IDMAX)*(4*MMAX)
DIMENSION X(1),ID(1),TEMP(1),ITEMP(1),NID(1)
ITEMP0=0
GO TO 20
ENTRY VERTEX (X,N,M,NMAX,MMAX,NID,IDMAX,TEMP,ITEMP)
C VERTEX ASSUMES POINTS WITH SAME ID ARE GROUPED TOGETHER IN X. NID
C IS A VECTOR OF LENGTH IDMAX HAVING IN NID(I) THE NO. OF PTS IN
C ITH GROUP.
ITEMP0=0
DO 10 NI=1,IDMAX
ITEMP0=ITEMP0+NID(NI)
IF(ITEMP0.NE.0) GO TO 20
10 CONTINUE
IF(ITEMP0.GT.N) STOP 12
GO TO 300
20 L=4*MMAX*IDMAX
DO 30 I=1,IDMAX
DO 30 J=1,2*M
30 ITEMPO(J+(I-1)*4*MMAX)=0
DO 200 I=1,N
I1=I
IF(ITEMP0.NE.0) GO TO 25
ID1=ID(I)
GO TO 38
25 IF(I.LE.ITEMPO)ID1=NI
IF(I.LE.ITEMPO) GO TO 38
NI=NI+1
IF(NI.GT.IDMAX) STOP 13
ITEMP0=ITEMP0+NID(NI)
ID1=NI
38 IM=2*M*(ID1-I)*4*MMAX
DO 100 J=1,M
II=J*(ID1-I)*4*MMAX
XI=X(I1+(J-1)*NMAX)
IF(ITEMP(I1).EQ.0) GO TO 50
IF(ITEMP(I1+M).EQ.0) GO TO 80
IF(ITEMP(IM).EQ.0) GO TO 100
40 IF(XIJ-TEMP(I1+L)) 60,42,50

```

```

42 GO 10 60
50 I0=ITEMP(I1)
   ITEMP(I1)=I1
   TEMP(I1+L)=XIJ
   I1=10
   IF(I1) 200,200,100
60 IF(XIJ-TEMP(I1+M+L)) 70,62,100
62 GO TO 100
70 I0=ITEMP(I1+M)
   ITEMP(I1+M)=I1
   TEMP(I1+M+L)=XIJ
   I1=10
   IF(I1) 200,200,100
80 IF(XIJ-TEMP(I1+L)) 70,100,90
90 TEMP(I1+M+L)=TEMP(I1+L)
   ITEMP(I1+M)=ITEMP(I1)
   ITEMP(I1)=I1
   TEMP(I1+L)=XIJ
   GO TO 200
100 CONTINUE
200 CONTINUE
300 RETURN
   END
CZXLPLP=====
SUBROUTINE ZX1LP (C,D,ICOLMS,ROW,K,M,N,ITMAX,LIC,IR,COPI
&,IDES,X,WA,IER)
C
C DIMENSION C(IR,1),COPI(IR,1),D(1),X(1),WA(1)
C DIMENSION ICOLMS(1),IDES(1),ROW(1)
C REAL C,D,ROW,COPI,X,WA,R1,R2,R3,D1,ZERO,ONE,
& EPS,D2,D3,T,RMT
C DATA EPS/1.0E-5/
C EPS IS USED IN TESTS FOR ZERO
C IF ABS(T) .LE. EPS, THEN T IS
C CONSIDERED TO BE ZERO
C ZERO/0.0/,ONE/1.0/
C
C DATA
C ISW = 3
C GO TO 5
C
C ENTRY ZX2LP (C,D,ICOLMS,ROW,K,M,N,ITMAX,LIC,IR,COPI,IDES,X,
& WA,IER)
C
C ISW = 2
C
C 5 IER = 0
C ITER = 0
C MP1 = M+1
C RMT = MP1*50
C DO 10 I=ISW,MP1
C   IF (D(I) GE. ZERO) GO TO 10
C   IER = 129
C   GO TO 9000
C 10 CONTINUE
C
C DO 20 I=1,MP1
C   D1 = ZERO
C   D2 = ZERO
C   DO 15 J=1,MP1
C     I = COPI(I,J)*D(J)
C     D1 = D1+T

```



```

15      D2 = D2*ABS(T)
      CONTINUE
      X(I) = D1
      D2 = D2*RT
      IF (D2*ABS(D1) .EQ. D2) D1 = ZERO
      IF (I.LT.ISW) GO TO 20
      IF (D1.LT.-EPS) IER = 130
20      CONTINUE
      IF (IER.NE.0) GO TO 9000
      FIND NEXT PIVOT COLUMN (IQ)

25      IQ = 0
      D3 = ZERO
      DO 26 I=1,MP1
26      D3 = D3*ABS(COPI(I,I))
      R1 = -EPS
      DO 35 J=1,N
      D1 = ZERO
      D2 = ZERO
      DO 30 I=1,MP1
      T = COPI(I,I)*C(I,J)
      D1 = D1*T
      D2 = D2*ABS(T)
30      CONTINUE
      D2 = D2*RT
      IF (D2*ABS(D1) .EQ. D2) GO TO 35
      IF (D1.GE.R1) GO TO 35
      R1 = D1
      IQ = J
35      CONTINUE
      DO 40 L=1,LIC
      II = ICOLMS(L)
      J = IABS(II)
      D1 = COPI(I,J)
      IF (II.LT.0) D1 = -D1
      IF (J.EQ.K) D1 = ZERO
      D2 = ONE
      IF (J.EQ.K) D2 = ZERO
      D1 = D1*COPI(I,K)*ROW(L)
      D2 = D2*ABS(ROW(L))
      D2 = D2*D3*RT
      IF (D2*ABS(D1) .EQ. D2) GO TO 40
      IF (D1.GE.R1) GO TO 40
      R1 = D1
      IQ = L*N
40      CONTINUE
      R3 = R1
      FIND NEXT PIVOT ROW (IP)

      R1 = -ONE
      IP = 0
      DO 65 I=1,MP1
      D1 = ZERO
      D3 = ZERO
      DO 45 J=1,MP1
      D1 = D1*COPI(I,J)*D(J)
      D3 = D3*ABS(COPI(I,J))
45      CONTINUE
      X(I) = D1
      IF (IO.GT.N) GO TO 55
      IF (IQ.EQ.0) GO TO 65
      D1 = ZERO

```



```

      &#x2D; ITERATIONS OF ALLOWED',ITMAX  

      GO TO 25  

95 IER = 132  

9000 CONTINUE  

      CALL UERTST (IER,'ZXILP' )  

9005 RETURN  

      END  

-----  

C ZX3LP  

      SUBROUTINE ZX3LP (A,IA,B,C,N,M1,M2,S,PSOL,DSOL,RW,IW,IER)  

C IER=130 MEANS IA=M1 + M2 +2  

131 COST CRITERION UNBOUNDED  

132 MAX_NUMBER OF ITERATIONS  

133 NO FEASIBLE SOLUTION EXISTS  

C C SOME ARTIFICIAL VARIABLES REMAINED IN SOLN BASIS AT  

C C 0 LEVEL AFTER PHASE 1 - MAYBE DUE TO REDUNDANT CON-  

C C STRAINTS. SOLN RETURNED IN PSOL & DSOL.  

C C  

      DIMENSION A(IA,1),B(1),C(1),PSOL(1),DSOL(1),RW(1),IW(1)  

      REAL A,B,C,PSOL,DSOL,RW,ZERO,TEMP,ONE,S,EPS  

      DATA EPS/1.0E-5/  

      EPS IS USED IN TESTS FOR ZERO  

      IF ABS(T) .LE. EPS, THEN T IS  

        CONSIDERED TO BE ZERO  

      DATA ZERO/0.0/,ONE/1.0/  

      ITMAX/2000/  

C  

      DATA  

      DATA  

      IER = 0  

      JER = 0  

      IEND = M1+M2  

      M12 = IEND  

C  

      IF (IA.GE.M12+2) GO TO 5  

      IER = 130  

      GO TO 9000  

5 M1P1 = M1+1  

      M1P2 = M1+2  

      M1P3 = M1+3  

      IEND1 = IEND+1  

C  

      DO 15 I=-1,IEND  

        K = IEND1-I  

        IR = K+2  

        B(K) = B(K)  

        PSOL(I) = I  

        DO 10 J=1,N  

          A(IR,J) = A(K,J)  

10 CONTINUE  

15 CONTINUE  

      IR = IEND+2  

C  

      IF (M2.EQ.0) GO TO 30  

      IBEG = M1+3  

      DO 25 I=IBEG,IR  

        IF (B(I).GE.ZERO) GO TO 25  

        B(I) = -B(I)  

        PSOL(I-2) = -PSOL(I-2)  

        DO 20 J=1,N  

          A(I,J) = -A(I,J)  

20 CONTINUE  

      C  

      CHECK EQUALITY CONSTRAINTS FOR  

      NEGATIVE RIGHT HAND SIDE.

```

```

25 CONTINUE
C
C
RE-ORDER OTHER CONSTRAINTS SO
B(I) .GE. 0 I = 1,...,M1-IQ
30 IQ = 0
IF (M1.EQ.0) GO TO 60
IEND = MIP2
INEXT = IEND
35 IF (B(IEND).GE.ZERO) GO TO 55
IF (INEXT.EQ.IEND) GO TO 45
TEMP = B(IEND)
B(IEND) = B(INEXT)
B(INEXT) = TEMP
TEMP = PSOL(IEND-2)
PSOL(IEND-2) = PSOL(INEXT-2)
PSOL(INEXT-2) = TEMP
DO 40 J=1,N
TEMP = A(IEND,J)
A(IEND,J) = A(INEXT,J)
A(INEXT,J) = TEMP
40 CONTINUE
45 IQ = IQ+1
PSOL(INEXT-2) = -PSOL(INEXT-2)
B(INEXT) = -B(INEXT)
DO 50 J=1,N
A(INEXT,J) = -A(INEXT,J)
50 CONTINUE
INEXT = INEXT + 1
55 IEND = IEND-1
IF (IEND.NE.2) GO TO 35
C
C
COMPUTE ROW 1 AND 2 OF A AND B
60 DO 65 J=1,N
A(2,J) = -C(J)
A(1,J) = ZERO
DO 65 I=2,IR
A(1,J) = A(1,J)-A(I,J)
65 CONTINUE
B(1) = ZERO
B(2) = ZERO
DO 70 I=3,IR
B(1) = B(1)-B(I)
70 CONTINUE
M = M12+1
IF (IA.EQ.IR) GO TO 80
C
C
PACK A
K = 0
L = 0
DO 75 J=1,N
DO 75 I=1,IR
K = MOD(K,IR+1)
IF (K.EQ.1) L = L+1
A(K,L) = A(I,J)
75 CONTINUE
C
C
GET ICOLMS AND ROW
80 LIC = IR+IQ
MIMIQ = M1 IQ
LI = MIP1-IQ
DO 95 I=1,LIC
IF (I.GE.MIP3) GO TO 90
IF (I GT.LI) GO TO 85
RW(I) = -ONE

```

```

      IW(1) = I+1
      GO TO 95
85    RW(1) = ONE
      IW(1) = -1-1
      GO TO 95
90    RW(1) = ZERO
      IW(1) = 1-IQ
95    CONTINUE
C
C
C
C
C
C
C
C
      WORK STORAGE ASSIGNMENTS
      ICOLMS(1) = IW(1)
      IDES(1) = IW(IDES)
      COPI(1,1) = RW(ICOP1)
      ROW(1) = RW(1)
      WA(1) = RW(IWK)
      X(1) = DSOL(1)
C
      GET IDES
C
      IW(MIP2) = 1
      IDES = LIC+1
      IW(IDES) = N+MIP2
      IDES2 = IDES+1
      IEND = IDES2+MIMIQ
      IENDIW = IDES2+M12
      K = N+1
      DO 100 I=IDES2,IENDIW
      IW(I) = K
      IF (I.EQ.IEND) K = N+MIP2
      K = K+1
100    CONTINUE
C
      GET COPI
C
      ICOP1 = IDES
      IWK = IR+IR+ICOP1
      K = IWK-1
      DO 105 I=ICOP1,K
      RW(I) = ZEPJ
105    CONTINUE
      L = ICOP1
      J = MIMIQ+1
      DO 110 I=1,J
      L = L+IR
      RW(L) = ONE
110    CONTINUE
      J = 0
      DO 115 I=ICOP1,K,IR
      RW(I+J) = ONE
      J = J+1
115    CONTINUE
      K = 1
C
      SOLVE PHASE 1 PROBLEM
      CALL ZX1LP (A,B,IW,RW,K,M,N,ITMAX,LIC,IR,RW(ICOP1),IW(IDES),
      &DSOL,RW(IWK),IER)
      IF (IER.NE.0) GO TO 185
      K = MIP2+N
      J = 2
C
C
C
      CHECK PHASE 1 SOLUTION FOR
      ARTIFICIAL VARIABLES
C
      DO 125 I=IDES2,IENDIW
      IF (IW(I).LE.K) GO TO 120
      IF (USOL(J).GT.EPS) GO TO 180
125    CONTINUE
      ARTIFICIAL VARIABLES REMAIN IN
      THE SOLUTION AT A ZERO LEVEL

```

```

      JER = 70
120 J = J+1
125 CONTINUE
C
C
      K = (N-1)*IR+1
      DO 140 L=1,K,IR
        J = (L+IA-1)/IA
        I = L-(J-1)*IA
        TEMP = A(I,J)
        IF (I.LT.IA) GO TO 130
        II = 1
        JJ = J+1
        GO TO 135
130 II = I+1
135 JJ = J
135 A(I,J) = A(II,JJ)
      A(II,JJ) = -TEMP
140 CONTINUE
C
C
      B(2) = -B(1)
      B(1) = ZERO
C
C
      J = ICOP1+IR-1
      DO 145 I=ICOP1,J
        TEMP = RW(I)
        RW(I) = RW(I+IR)
        RW(I+IR) = TEMP
145 CONTINUE
      J = ICOP1+IR+IR-1
      DO 150 I=ICOP1,J,IR
        TEMP = KW(I)
        RW(I) = RW(I+1)
        RW(I+1) = TEMP
150 CONTINUE
      K = 2
C
      DO 155 I=1,LIC
        RW(I) = -RW(I)
155 CONTINUE
C
      IW(1) = 1
C
      J = IW(IDES)
      IW(IDES) = IW(IDES2)
      IW(IDES2) = J
      IW(MIP2) = -2
      LICSV = LIC
C
      IF (JER.EQ.0) LIC = MIP2
C
      CALL ZXILP (A,B,IW,RW,K,M,N,ITMAX,LIC,IR,RW(ICOP1),IW(IDES),
        & DSOL,RW(IWK),ITER)
      LIC = LICSV
      S = DSOL(1)
C
      RE-ORDER THE PRIMAL SOLUTION

```

```

DO 160 J=1,M12
  RW(J) = PSOL(J)
160 CONTINUE
DO 165 J=1,N
  PSOL(J) = ZERO
165 CONTINUE
DO 170 J=1,M
  K = IW(IDES+J)
  IF (K.GT.N) GO TO 170
  PSOL(K) = OSOL(J+1)
170 CONTINUE
C
      GET DUAL SOLUTION
      JI = LIC+1+IR
      TEMP = RW(JI)
      DO 175 I=1,M12
        J = ABS(RW(I))
        JI = JI+IR
        OSOL(J) = RW(JI)+TEMP
        IF (RW(I).LT.ZERO) OSOL(J) = -OSOL(J)
        IF (I.LE.M1) OSOL(J) = ABS(OSOL(J))
175 CONTINUE
      GO TO 195
C
      ARTIFICIAL VARIABLES ARE IN THE
      SOLUTION HENCE NO FEASIBLE
      SOLUTION EXISTS
C
      RESTORE A AND B
      UNPACK A
      DO 200 II=1,IR
        A(I,J) = A(K,L)
        I = I-1
        K = K-1
        IF (K.NE.0) GO TO 200
        K = IA
        L = L-1
200 CONTINUE
        J = J-1
205 CONTINUE
210 DO 220 I=1,M12
      B(I) = B(I+2)
      DO 215 J=1,N
        A(I,J) = A(I+2,J)
215 CONTINUE
220 CONTINUE
C
      PERMUTE ROWS OF A AND ELEMENTS OF B
      ACCORDING TO PERMUTATIONS STORED IN
      RW
      DO 230 I=1,M12
        J = ABS(RW(I))
        IF (J.EQ.1) GO TO 230
        TEMP = RW(I)

```

```

RW(I) = RW(J)
RW(J) = TEMP
TEMP = B(I)
B(I) = B(J)
B(J) = TEMP
DO 225 K=1,N
  TEMP = A(I,K)
  A(I,K) = A(J,K)
  A(J,K) = TEMP
225 CONTINUE
230 CONTINUE
DO 240 I=1,M12
  IF (RW(I).GT..ZERO) GO TO 240
  B(I) = -B(I)
DO 235 J=1,N
  A(I,J) = -A(I,J)
235 CONTINUE
240 CONTINUE
IF (IER.EQ. 0) IER=JER
9000 CONTINUE
IF (JER.NE.0) CALL UERTST (JER,'ZX3LP')
IF (IER.GT.JER) CALL UERTST (IER,'ZX3LP')
9005 RETURN
END
C DISCRM=====
SUBROUTINE DISCRM(X,TD,K,H,NV0,NP,NO,P,D1,D2,SOS,IER)
C DISCRM SEEKS HYPERPLANES DESCRIBED BY VECTOR P, SCALARS D1,D2. FOR
C POINT Z, Z=P GE U1 OR LE D2 AS ID IS K OR H. BY REQUIRING D2-D1
C GE 0 FOR OVERLAPPING SETS (SOS=1), LE 0 FOR SEPARATED SETS (SOS=-1)
C MAXIMIZING D1 D2 ASSURES EITHER MIN OVERLAP OR MAX SEPARATION.
C P,D1,D2 ARE SCALED SO THAT P HAS UNIT LENGTH.
C NO IS NUMBER OF PTS IN THE TWO SETS, X IS AN NP*NVO ARRAY WITHIN
C AN NPMAX*NVMAX ARRAY. WK IS WORK VECTOR OF LENGTH LB, A LITTLE LT
C (NV0+3)*(NO+1)+2*(NO+5)**2 IF NO GE NV0.
C IER=0 IF SUCCESS; =33,34,70 IF WARNING FROM SV DECOMP OR LP
C ROUTINES; =129 IF SV DECOMP FAILS, =130-133 IF LP FAILS, =135 IF
C DIMENSIONS WRONG.
C
PARAMETER (NPMAX=450, NVMAX=14, ICLMAX=14)
PARAMETER (NAXES=NVMAX*ICLMAX)
COMMON LNPMAX,LNVMAX,LICLMAX,LNAXES,LLWK,WK
INTEGER H
DIMENSION ID(1),WK(1),P(1)
DIMENSION A(1),V(1),W(1),U(1),Q(1),TEMP(1)
DIMENSION B(1),C(1),PSOL(1),DSOL(1),RW(1),IW(1)
EQUIVALENCE (A(1),WK(1)),(V(1),WK(1)),(W(1),WK(1))
&(U(1),WK(1)),(Q(1),WK(1)),(TEMP(1),WK(1)),(B(1),WK(1)),
&(C(1),WK(1)),(PSOL(1),WK(1)),(DSOL(1),WK(1)),(IW(1),WK(1)),
&(RW(1),WK(1))
N=NO+1
NV=NV0+2
M=MAX0(N,NV)
LU=M*N
L1=L0+1
L2=L1+(N+3)*N
L3=L2+3*MAX0(N,NV)
L4=L3+N
IF (LLWK.GE.L3+NV+NV+1) GO TO 5
WRITE(06,*) ' IN DISCRM, WK REQUIRES MORE THAN L3+(NO+1)*(NV0+1)
&+(NV0+1)+1 =',L3+N*NV+NV+1,' WORDS BUT HAS AVAILABLE ONLY LLWK

```



```

      8, ' , LLWK
      IER=135
      GO TO 600
5 CONTINUE
      IER=0
      TOL=.001
      IF(NPMAX.EQ.LNPMAX) GO TO 10
      WRITE(06,*) ' IN DISCRIM NPMAX=', NPMAX, ' BUT IN MAIN NPMAX ='
      8, LNPMAX
      IER=135
      GO TO 600
10 IF(NVMAX.EQ.LNVMAX) GO TO 15
      WRITE(06,*) ' IN DISCRIM NVMAX=', NVMAX, ' BUT IN MAIN NVMAX='
      8, LNVMAX
      IER=135
      GO TO 600
15 L=0
      DO 80 I=1,NP
      READ(22)(WK(L4+J), J=' , NVU)
      IF(ID(I).EQ.K) GO TO 20
      IF(ID(I).EQ.H) GO TO 30
      GO TO 80
20 TEMP1=+1.
      TEMP2=-1.
      TEMP3=0.
      GO TO 40
30 TEMP1=-1.
      TEMP2=0.
      TEMP3=+1.
40 L=L+1
      IF(L.LE.NO) GO TO 45
      WRITE(06,*) ' DISCRIM COUNTS MORE THAN', NO, ' POINTS WITH ID'
      8, K, ' OR', H
      IER=136
      GO TO 600
45 DO 50 J=1,NVO
50 A(L+(J-1)*M)=TEMP1*WK(L4+J)
      A(L+NVO*M)=TEMP2
      A(L+(NVO+1)*M)=TEMP3
80 CONTINUE
      DO 85 J=1,NVO
85 A(L+1+(J-1)*M)=0.
      A(L+1+NVO*M)=-SOS
      A(L+1+(NVO+1)*M)=+SOS
C GET SING VALUE DECOMPOSITION OF N*NV MATRIX A AS U*Q*VTRANSPOSE
C WHERE U IS N*N (BUT IMBEDDED IN AN (N*3)*N ARRAY), Q IS N*NV,
C V IS NV**2. ONLY THE NV DIAGONAL WORDS OF Q ARE STORED. V IS
C STORED OVER A. U IS INITIALIZED TO IDENTITY. UTRANSPOSE STORED
C OVER U. TEMP IS 2NV WORDS LONG
      DO 120 I=1,N
      DO 110 J=1,N
110 U(L+1+(J-1)*M)=0.
120 U(L+1+(I-1)*M)=1.
      CALL LSVDF(A,M,N,NV,U(L+1),N+3,N,Q(L2+1),TEMP(L2+NVO+1),IER)
      IF(IER.EQ.33.OR.IER.EQ.34)WRITE(06,*) ' WARNING: LSVDF IN
      8 DISCRIM RETURNS IER=', IER
      IF(IER.EQ.129) GO TO 600
      QMAX=0

```

```

DO 130 I=1,NV
  IF (ABS(Q(L2+1)).GT.QMAX)QMAX=ABS(Q(L2+1))
130 CONTINUE
DO 140 I=1,NV
  IF (ABS(Q(L2+1)).LT.TOL*QMAX)Q(L2+1)=0.
140 CONTINUE
C REPLACE V BY W=VDQTUT, AN NV*N MATRIX. T=TRANPOSE. D IS NV**2
C GENERALIZED INVERSE OF QTQ.
DO 145 I=1.(N+1)*(NV-1)+1
145 RW(L3+1+((I-1)/M)*(NV-M))=V(I)
DO 160 I=1,NV
DO 150 J=1,N
  W(I+(J-1)*NV)=0.
DO 150 L=1,NV
  IF(Q(L2+L).EQ.0.) GO TO 150
  W(I+(J-1)*NV)=W(I+(J-1)*NV)+RW(L3+I+(L-1)*NV)*
    &U(L1+L*(J-1)*(N+3))/Q(L2+L)
150 CONTINUE
160 CONTINUE
C SET UP LP PROBLEM: (D-1)UT*PSOL=B, SUM PSOL LE 1, PSOL GE 0.
C MAXIMIZE S=C*PSOL. IN CODE, INEQ CONSTRAINT AFFIXED AT BEGINNING
C OF U, AS ROW.
C SET UP LP PROBLEM: (D-1)UT*PSOL=B, MAXIMIZE S=C*PSOL, PSOL GE 0
C D IS DIAG MATRIX WITH II ENTRY 1.0 AS II ENTRY OF Q IS NE.EQ 0.
DO 165 J=1,N0
165 U(L0+1+(J-1)*(N+3))=-SOS
  U(L0+1+(N-1)*
    &(N+3))=0.
  ICONST=0
DO 180 I=1,N
  IF(I.LE.NV.AND.Q(L2+I).NE.0.) GO TO 180
  ICONST=ICONSTR+1
DO 170 J=1,N
170 U(L1+ICONSTR*(J-1)*(N+3))=U(L1+I+(J-1)*(N+3))
180 CONTINUE
  B(L2+1)=-SOS
DO 175 I=2,ICONSTR+1
175 B(L2+I)=0.
DO 210 J=1,N0
210 C(L3+J)=0.
  C(L3+N)=-SOS
  L5=L4*MAX0(N,ICONSTR+2)
  L6=L5+ICONSTR+4
  L7=L6+2*ICONSTR+10
  L8=L7+(ICONSTR+5)**2+1
  IF(LLWK.GE.L8) GO TO 190
  IER=135
  WRITE(06,*)' IN DISCRIM, WK REQUIRES LB=',LB,' WORDS BUT HAS
    & AVAILABLE ONLY LLWK=' ,LLWK
  GO TO 600
190 CALL ZX3LP(U(L0+1),N+3,B(L2+1),C(L3+1),N,1,ICONSTR,S,
  &PSOL(L4+1),DSOL(L5+1),RW(L7+1),IW(L6+1),IER)
  IF(IER.EQ.133) GO TO 600
C COMPUTE P AND D1,D2
DO 230 I=1,NV0
  P(I)=0.
DO 230 J=1,N
  IF (ABS(PSOL(L4+J)).LT.1.E30) GO TO 230
  IER=137
  GO TO 600

```

```

230 P(I)=P(I)+W(I+(J-1)*NV)*PSOL(L4+J)
D1=0.
D2=0.
DO 240 J=1,N
D1=D1+W(NV0+1+(J-1)*NV)*PSOL(L4+J)
240 D2=D2+W(NV0+2+(J-1)*NV)*PSOL(L4+J)
C NORMALIZE P TO UNIT LENGTH1, ADJUST D1,D2
TEMPO=0.
DO 250 I=1,NV0
250 TEMPO=TEMPO+P(I)**2
IF(TEMPO.NE.0) GO TO 245
IER=IER+200
GO TO 600
245 TEMPO=SQRT(TEMPO)
DO 260 I=1,NV0
260 P(I)=P(I)/TEMPO
D1=D1/TEMPO
D2=D2/TEMPO
600 REWIND 22
RETURN
END
CMESSAGE=====
SUBROUTINE MESSAGE(K)
C PRINTS KTH MESSAGE
GO TO (1,2,3,4,5,6,7,8,9).K
1 WRITE(06,*) ' GALACTIC GEOMETRIC ANALYSIS OF LARGE ARRAYS
& CONTAINING >3 INDEP COORD'.
RETURN
2 WRITE(06,*) ' ENTER NUMBER & NAMES OF SAMPLE FILES,WORDS/
& RECORD.V/N TO CLEARFILES'.
RETURN
3 WRITE(06,*) ' WHICH OTHER VARIABLE IDENTIFIES SUBSETS (0 IF
& NONE)?'.
RETURN
4 WRITE(06,*) ' ENTER N (LE 20) THEN NAME N POINTS TO BE
& EXCLUDED'.
RETURN
5 WRITE(06,*) ' ENTER N, THEN N PAIRS OF IDS TO BE LABELED AS
& THE FIRST'.
RETURN
6 WRITE(06,*) ' FOR POINTS ON INPUT FILE, NOT EXCLUDED BY USER'.
RETURN
7 WRITE(06,*) ' CHOOSE TYPICAL,SCALE OPTIONS(MIN,MAX,AVER,SIGMA
& MID,RANGE,SPECIAL).
RETURN
8 WRITE(06,*) ' VAR MIN MID MAX RANGE
& AVER SIGMA'.
RETURN
9 WRITE(06,*) ' ENTER COUNT OF BONDED POINTS(I&J FORCED INTO SAME
& CLUSTER).
RETURN
END
CATFILE=====
SUBROUTINE ATFILE (FILIN,I)
C SETS UP FILIN(I) AS FILE 01, IF FILIN(I) IS 0, RETURNS I=0.
CHARACTER *64FILIN(1),FILECR
FILECR=
IF(I.GT.1) REWIND 01
CLOSE(01,STATUS='KEEP')
IF(I.GT.0) GO TO 10

```

```

WRITE(06,*) ' STOP. AFILE CALLED WITH FILE NUMBER LE 0'
STOP
10 IF (FILIN(1).NE.FILECR) GO TO 20
IF (1.LE.1) REWIND 01
I=0
RETURN
20 OPEN(01,FILE=FILIN(1),STATUS='OLD',IOSTAT=ISTAT,
&ACCESS='SEQUENTIAL',FORM='FORMATTED')
IF (IOSTAT.NE.0) THEN
WRITE(06,*) ' ERROR IN SUBROUTINE AFILE OPENING FILE',
& I, ISTAT, ISTAT
END IF
RETURN
END
CLSVDB=====
SUBROUTINE LSVDB (D,E,N,V,IV,NRV,C,IC,NCC,IER)
C SINGULAR VALUE DECOMPOSITION OF A BIDIAGONAL MATRIX (IMSL)
INTEGER N,IV,NRV,IC,NCC,IER
REAL D(N),E(N),V(IV,1),C(IC,1)
I,II,J,K,KK,L,LL,LP1,NORS,NIO
WNTV,HAVERS,FAIL
DNORM,ZERO,ONE,TWO,CS,F,SQINF,FTEMP,G,H,HTEMP,
& SN,T,X,Y,Z
SQINF=0.13043818E+20/
ZERO=0.0/ONE=1.0/TWO=2.0/
FIRST EXECUTABLE STATEMENT
IER = 0
IF (N.LE.0) GO TO 9005
NIO = 10*N
WNTV = NRV,GT.0
HAVERS = NCC,GT.0
FAIL = .FALSE.
NORS = 0
E(1) = ZERO
DNORM = ZERO
DO 5 J=1,N
5 DNORM = AMAX1(ABS(D(J))+ABS(E(J)),DNORM)
DO 100 KK=1,N
K = N+1-KK
TEST FOR SPLITTING OR RANK
DEFICIENCIES FIRST MAKE TEST FOR
LAST DIAGONAL TERM, D(K), BEING
SMALL.
10 IF (K.EQ.1) GO TO 25
T = DNORM+D(K)
IF (T.NE.DNORM) GO TO 25
SINCE D(K) IS SMALL WE WILL MAKE A
SPECIAL PASS TO TRANSFORM E(K) TO
ZERO.
CS = ZERO
SN = -ONE
DO 20 II=2,K
I = K+1-II
F = -SN*E(I+1)
E(I+1) = CS*E(I+1)
T = D(I)
FTEMP = F
CALL SROTG (D(I),FTEMP,CS,SN)
TRANSFORMATION CONSTRUCTED TO ZERO

```

```

C          IF (.NOT. WNTV) GO TO 20
C          DO 15 J=1, NRV
C          CALL LSVG2 (CS, SN, V(J,1), V(J,K))
15
C          CONTINUE
C          ACCUMULATE RT. TRANSFORMATIONS IN V.
C          THE MATRIX IS NOW BIDIAGONAL, AND OF
C          LOWER ORDER SINCE E(K) .EQ. ZERO
25      DO 30 LL=1,K
          L = K+1-LL
          T = DNORM+E(L)
          IF (T.EQ.DNORM) GO TO 50
          T = DNORM+D(L-1)
          IF (T.EQ.DNORM) GO TO 35
30      CONTINUE
C          THIS LOOP CANT COMPLETE SINCE E(1) =
C          ZERO.
C          CANCELLATION OF E(L), L.GT.1.
35      CS = ZERO
          SN = -ONE
          DO 45 I=L,K
              F = -SN+E(I)
              E(I) = CS+E(I)
              T = DNORM+F
              IF (T.EQ.DNORM) GO TO 50
              T = D(I)
              FTEMP = F
              CALL SROTG (D(I), FTEMP, CS, SN)
              IF (.NOT. HAVES) GO TO 45
              DO 40 J=1, NCC
                  CALL LSVG2 (CS, SN, C(I,J), C(L-1,J))
40      CONTINUE
45      TEST FOR CONVERGENCE
          Z = D(K)
          IF (L.EQ.K) GO TO 85
          SHIFT FROM BOTTOM 2 BY 2 MINOR OF
          B**(T)*B.
          X = D(L)
          Y = D(K-1)
          G = E(K-1)
          H = E(K)
          F = ((V-Z)*(Y+Z)+(G-H)*(G+H))/((TWO*H*Y)
          G = ABS(F)
          IF (ABS(F) .LT. SQINF) G = SQRT(ONE+F**2)
          IF (F.LT.ZERO) GO TO 55
          T = F+G
          GO TO 60
55      T = F-G
          F = ((X-Z)*(X+Z)+H*(Y/T-H))/X
          GO TO 60
          NEXT QR SWEEP
          CS = ONE
          SN = ONE
          LP1 = L+1
          DO 80 I=LP1,K
              G = E(I)
              Y = D(I)
              H = SN*G
              G = CS*G
80

```

```

      HTEMP = H
      CALL SROTG (F,HTEMP,CS,SN)
      E(I-1) = F
      F = X*CS+G*SN
      G = -X*SN+G*CS
      H = V*SN
      V = V*CS
      IF (.NOT.W*Z) GO TO 70
      ACCUMULATE ROTATIONS (FROM THE
      RIGHT) IN V
      DO 65 J=1,NR
      CALL LSVG2 (CS,SN,V(J,I-1),V(J,I))
      HTEMP = H
      CALL SROTG (F,HTEMP,CS,SN)
      D(I-1) = F
      F = CS*G+SN*V
      V = -SN*G+CS*V
      IF (.NOT.HAVERS) GO TO 80
      DO 75 J=1,NJC
      CALL LSVG2 (CS,SN,C(I-1,J),C(I,J))
      APPLY ROTATIONS FROM THE LEFT TO
      RIGHT HAND SIDES IN C
      CONTINUE
      F(L) = ZERG
      E(K) = F
      D(K) = X
      NQRS = NQRS+1
      IF (NQRS.LE.NIG) GO TO 10
      RETURN TO TEST FOR SPLITTING.
      FAIL = .TRUE.
      CUTOFF FOR CONVERGENCE FAILURE. NQRS
      WILL BE 2*N USUALLY.
      DO 85 I=2,N
      IF (Z.GE.ZERO) GO TO 95
      D(K) = -Z
      IF (.NOT.WNTV) GO TO 95
      DO 90 J=1,NRV
      V(J,K) = -V(J,K)
      CONTINUE
      CONVERGENCE. D(K) IS MADE
      NONNEGATIVE
      DO 100 CONTINUE
      IF (N.EQ.1) GO TO 140
      DO 105 I=2,N
      IF (D(I).GT.D(I-1)) GO TO 110
      CONTINUE
      GO TO 140
      EVERY SINGULAR VALUE IS IN ORDER
      DO 110 DO 135 I=2,N
      T = D(I-1)
      K = I-1
      DO 115 J=I,N
      IF (.GE.D(J)) GO TO 115
      T = D(J)
      K = J
      CONTINUE
      IF (K.EQ.I-1) GO TO 135
      D(K) = D(I-1)
      D(I-1) = T
      IF (.NOT.HAVERS) GO TO 125

```

```

DO 120 J=1,NCG
  T = C(I-1,J)
  C(I-1,J) = C(K,J)
120  C(K,J) = T
125  IF (.NOT.WNTV) GO TO 135
DO 130 J=1,NRV
  T = V(J,I-1)
  V(J,I-1) = V(J,K)
130  V(J,K) = T
135  CONTINUE
      END OF ORDERING ALGORITHM.
C
140  IER = 129
  IF (FAIL) GO TO 9000
      CHECK FOR POSSIBLE RANK DEFICIENCY
C
  IER = 33
  T = 0.0
  IF (D(1).NE.ZERO) T=D(N)/D(1)
  F=100.0+T
  IF (F.EQ.100.0) GO TO 9000
  IER = 0
  GO TO 9005
9000  CONTINUE
  CALL UERTST (IER,'LSVDB ')
9005  RETURN
  END
CLSVDF=====
C
  SUBROUTINE LSVDF (A,IA,M,N,B,IB,NB,S,WK,IER)
C SINGULAR VALUE DECOMPOSITION OF A REAL MATRIX (IMSL)
C IER=33 MEANS MATRIX A IS NOT OF FULL RANK OR VERY ILLCONDITIONED
C SMALL SINGULAR VALUES MAY NOT BE VERY ACCURATE.
C 34 EITHER N.LE.0 OR M.LE.0
C 129 CONVERGENCE WAS NOT OBTAINED BY LSVDB. COMPUTATION
C WAS DISCONTINUED.
      IA,M,N,IB,NB,IER
      REAL A(IA,N),B(IB,1),S(N),WK(N,2)
      INTEGER I,J,JP1,K,L,MM,NN,NNP1,NS,NSP1
      REAL ZERO,ONE,T
      DATA ZERO/0.0/,ONE/1.0/
      FIRST EXECUTABLE STATEMENT
C
  IER = 0
C BEGIN SPECIAL FOR ZERO ROWS & COLS. PACK THE NONZERO
C COLUMNS TO THE LEFT
  NN = N
  IER = 34
  IF (NN.LE.0.OR.M.LE.0) GO TO 9000
  IER = 0
  J = NN
  5  CONTINUE
  DO 10 I=1,M
    IF (A(I,J).NE.ZERO) GO TO 25
  10  CONTINUE
      COL J IS ZERO. EXCHANGE IT WITH COL
      N
  IF (J.EQ.NN) GO TO 20
  DO 15 I=1,M
  15  A(I,J) = A(I,NN)
  20  CONTINUE
  A(1,NN) = J
  NN = NN-1

```

```

25 CONTINUE
J = J-1
IF (J.GE.1) GO TO 5
IF N=0 THEN A IS ENTIRELY ZERO AND
SVD COMPUTATION CAN BE SKIPPED
C
C
NS = 0
IF (NN.EQ.0) GO TO 120
PACK NONZERO ROWS TO THE TOP QUIT
PACKING IF FIND N NONZERO ROWS
C
C
I = 1
MM = M
30 IF (I.GT.N.OR.I.GE.MM) GO TO 75
IF (A(I,I).NE.ZERO) GO TO 40
DO 35 J=1,NN
IF (A(I,J).NE.ZERO) GO TO 40
35 CONTINUE
GO TO 45
40 I = I+1
GO TO 30
C
C
ROW I IS ZERO EXCHANGE ROWS I AND M
45 IF (NB.LE.0) GO TO 55
DO 50 J=1,NB
T = B(I,J)
B(I,J) = B(MM,J)
B(MM,J) = T
50 CONTINUE
55 DO 60 J=1,NN
60 A(I,J) = A(MM,J)
IF (MM.GT.NN) GO TO 70
DO 65 J=1,NN
65 A(MM,J) = ZERO
70 CONTINUE
C
C
EXCHANGE IS FINISHED
MM = MM-1
GO TO 30
C
75 CONTINUE
C
C
END SPECIAL FOR ZERO ROWS AND
COLUMNS
BEGIN SVD ALGORITHM..
(1) REDUCE THE MATRIX TO UPPER
BIDIAGONAL FORM WITH HOUSEHOLDER
TRANSFORMATIONS.
H(N)...H(1)AQ(1)...Q(N-2) =
(D**T,0)**T WHERE D IS UPPER
BIDIAGONAL.
(2) APPLY H(N)...H(1) TO B. HERE
H(N)...H(1)*B REPLACES B IN
STORAGE.
(3) THE MATRIX PRODUCT W=
Q(1)...Q(N-2) OVERWRITES THE FIRST
N ROWS OF A IN STORAGE.
(4) AN SVD FOR D IS COMPUTED. HERE K
ROTATIONS RI AND PI ARE COMPUTED
SO THAT RK...RI*D*PI**(T)...PK**(T)
= DIAG(S1,...,SM) TO WORKING
ACCURACY. THE S1 ARE NONNEGATIVE
AND NONINCREASING. HERE RK...RI*B
OVERWRITES B IN STORAGE WHILE
A*PI**(T)...PK**(T) OVERWRITES A

```



```

C
C      IN STORAGE.
C      (5) IT FOLLOWS THAT, WITH THE PROPER
C      DEFINITIONS, U*(1)*B OVERWRITES
C      B, WHILE V OVERWRITES THE FIRST N
C      ROW AND COLUMNS OF A.
C
C      L = MINO(MM,NN)
C
C      THE FOLLOWING LOOP REDUCES A TO
C      UPPER BIDIAGONAL AND ALSO APPLIES
C      THE PREMULPLYING TRANSFORMATIONS
C      TO B.
C
C      DO 85 J=1,L
C        IF (J.GE.MM) GO TO 80
C        JP1 = MINO(J+1,NN)
C        CALL VHS12 (1,J,J+1,MM,A(1,J),1,T,A(1,JP1),1,IA,NN-J)
C        CALL VHS12 (2,J,J+1,MM,A(1,J),1,T,B,1,IB,NB)
C      80 IF (J.GE.NN-1) GO TO 85
C        CALL VHS12 (1,J+1,J+2,NN,A(J,1),IA,WK(J,2),A(J+1,1),IA,1,MM-J)
C      85 CONTINUE
C
C      COPY THE BIDIAGONAL MATRIX INTO THE
C      ARRAY S FOR LSVDB
C
C      IF (L.EQ.1) GO TO 95
C      DO 90 J=2,L
C        S(J) = A(J,J)
C        WK(J,1) = A(J-1,J)
C      90 CONTINUE
C      95 S(1) = A(1,1)
C
C      NS = NN
C      IF (MM.GE.NN) GO TO 100
C      NS = MM+1
C      S(NS) = ZERO
C      WK(NS,1) = A(MM,MM+1)
C      100 CONTINUE
C
C      CONSTRUCT THE EXPLICIT N BY N
C      PRODUCT MATRIX, W=Q1*Q2*...*QL*1
C      IN THE ARRAY A
C
C      DO 115 K=1,NN
C        I = NN+1-K
C        IF (I.GT.MINO(MM,NN-2)) GO TO 105
C        CALL VHS12 (2,I+1,I+2,NN,A(I,1),IA,WK(I,2),A(I+1,1),IA,NN-I)
C      105 DO 110 J=1,NN
C        A(I,J) = ZERO
C      110 A(I,I) = ONE
C      115 CONTINUE
C
C      COMPUTE THE SVD OF THE BIDIAGONAL
C      MATRIX
C
C      LEVEL=1
C      CALL UERSET(LEVEL,LEVOLD)
C      CALL LSVDB (S(1),WK(1,1),NS,A,IA,NN,B,IB,NB,IER)
C      TEST FOR IER=33
C
C      IF (IER.GT.128) GO TO 9000
C      CALL UERSET(LEVEL,LEVOLD)
C      IF (IER.NE.33) GO TO 120
C      T=U.U
C      NM=MINO(M,N)
C      IF (S(1).NE.ZERO) T=S(NM)/S(1)
C      F=100.0*T
C      IF (F.EQ.100.0) GO TO 120

```

```

IER=U
120 CONTINUE
IF (NS.LT. MIN0(M,N)) IER = 33
IF (NS.GE.NN) GO TO 130
NSP1 = NS+1
DO 125 J=NSP1,NN
125 SCJ = ZERO
130 CONTINUE
IF (NN.EQ.N) GO TO 155
NNP1 = NN+1
C
C      MOVE RECORD OF PERMUTATIONS AND
      STORE ZEROS
DO 140 J=NNP1,N
S(J) = A(I,J)
IF (NN.LT.1) GO TO 140
DO 135 I=1,NN
A(I,J) = ZERO
140 CONTINUE
C
C      PERMUTE ROWS AND SET ZERO SINGULAR
      VALUES
DO 150 K=NNP1,N
I = S(K)
S(K) = ZERO
DO 145 J=1,N
A(K,J) = A(I,J)
145 A(I,J) = ZERO
A(I,K) = ONE
150 CONTINUE
C
C      END SPECIAL FOR ZERO ROWS AND
      COLUMNS
155 IF (IER.EQ.0) GO TO 9005
9000 CONTINUE
CALL UERIST (IER, 'LSVDF ')
9005 RETURN
END
CLSVG1=====
C
SUBROUTINE LSVG1 (A,B,CS,SN,SIG)
C NUCLEUS CALLED ONLY BY IMSL ROUTINE LS,DB
REAL
A,B,CS,SN,SIG
AA,BB
IF (ABS(A).LE.ABS(B)) GO TO 5
AA = ABS(A+A)
SIG = AA*SQRT(0.25*(B/AA)**2)
CS = A/SIG
SN = B/SIG
RETURN
5 IF (B.EQ.0.0) GO TO 10
BB = ABS(B+B)
SIG = BB*SQRT(0.25*(A/BB)**2)
CS = A/SIG
SN = B/SIG
RETURN
10 SIG = 0.0
CS = U.U
SN = 1.0
RETURN
END
C IMSL ROUTINE NAME - LSVG2
C

```

```

CLSVG2=====
C
C SUBROUTINE CLSVG2 (LS,SN,X,Y)
C
C REAL LS,SN,X,Y
C REAL LS,SN,X,Y
C XR
C XR=CS*X+SN*Y
C Y=-SN*X+LS*Y
C X=XR
C RETURN
C END
C
CUERSET=====
SUBROUTINE UERSET (LEVEL,LEVOLD)
C
C INTEGER LEVEL,LEVOLD
C LEVOLD = LEVEL
C CALL UERTST (LEVOLD,'UERSET')
C RETURN
C END
C
CUERTST=====
SUBROUTINE UERTST (IER,NAME)
C
C INTEGER IER
C CHARACTER*6 NAME
C CHARACTER*6 NAME
C INTEGER I,IEQ,IEQDF,IOUNT,LEVEL,LEVOLD,NAMEQ(6),
C NAMESET(6),NAMUPK(6),NIN,NMTB
C NAMESET/1HU,1HE,1HR,1HS,1HE,1HT/
C NAMEQ/6*1H /
C LEVEL/4/,IEQDF/0/,IEQ/1H=/
C UNPACK NAME INTO NAMUPK
C CALL USPXD (NAME,6,NAMUPK,NMTB)
C CALL UGETIO(1,NIN,IOUNT)
C GET OUTPUT UNIT NUMBER
C CHECK IER
C IF (IER.GT.999) GO TO 25
C IF (IER.LT.-32) GO TO 55
C IF (IER.LE.128) GO TO 5
C IF (LEVEL.LT.1) GO TO 30
C PRINT TERMINAL MESSAGE
C IF (IEQDF.EQ.1) WRITE(IOUNT,35) IER,NAMEQ,IEQ,NAMUPK
C IF (IEQDF.EQ.0) WRITE(IOUNT,35) IER,NAMUPK
C GO TO 30
C 5 IF (IER.LE.64) GO TO 10
C IF (LEVEL.LT.2) GO TO 30
C PRINT WARNING WITH FIX MESSAGE
C IF (IEQDF.EQ.1) WRITE(IOUNT,40) IER,NAMEQ,IEQ,NAMUPK
C IF (IEQDF.EQ.0) WRITE(IOUNT,40) IER,NAMUPK
C GO TO 30
C 10 IF (IER.LE.32) GO TO 15
C PRINT WARNING MESSAGE
C IF (LEVEL.LT.3) GO TO 30
C IF (IEQDF.EQ.1) WRITE(IOUNT,45) IER,NAMEQ,IEQ,NAMUPK
C IF (IEQDF.EQ.0) WRITE(IOUNT,45) IER,NAMUPK
C GO TO 30
C 15 CONTINUE
C CHECK FOR UERSET CALL
C DO 20 I=1,6
C IF (NAMUPK(I).NE.NAMESET(I)) GO TO 25
C 20 CONTINUE
C LEVOLD = LEVEL
C IER = IER
C IER = LEVOLD

```

```

IF (LEVEL.LT.0) LEVEL = 4
IF (LEVEL.GT.4) LEVEL = 4
GO TO 30
25 CONTINUE
IF (LEVEL.LT.4) JC TO 30
C PRINT NON-DEFINED MESSAGE
IF (IEQDF.EQ.1) WRITE(IUNIT,50) IER,NAMEQ,IEQ,NAMUPK
IF (IEQDF.EQ.0) WRITE(IUNIT,50) IER,NAMUPK
30 IEQDF = 0
RETURN
35 FORMAT(19H *** TERMINAL ERROR,10X,7H(IE = ,I3,
& 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)
40 FORMAT(27H *** WARNING WITH FIX ERROR,2X,7H(IE = ,I3,
& 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)
45 FORMAT(18H *** WARNING ERROR,11X,7H(IE = ,I3,
& 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)
50 FORMAT(20H *** UNDEFINED ERROR,9X,7H(IE = ,I5,
& 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)
C
C SAVE P FOR P = R CASE
C P IS THE PAGE NAMUPK
C R IS THE ROUTINE NAMUPK
55 IEQDF = 1
DO 60 I=1,6
60 NAMEQ(I) = NAMUPK(I)
65 RETURN
END
CUGETIO=====
SUBROUTINE UGETIO(IOPT,NIN,NOUT)
INTEGER IOPT,NIN,NOUT
INTEGER NIND,NOUTD
DATA NIND/5/,NOUTD/6/
IF (IOPT.EQ.3) GO TO 10
IF (IOPT.EQ.2) GO TO 5
IF (IOPT.NE.1) GO TO 9005
NIN = NIND
NOUT = NOUTD
GO TO 9005
5 NIND = NIN
GO TO 9005
10 NOUTD = NOUT
9005 RETURN
END
CUSPKD=====
SUBROUTINE USPKD (PACKED,NCHARS,UNPAKD,NCHMTB)
C NUCLEUS CALLED BY IMSL NC,NCHARS,NCHMTB
C INTEGER
C UNPAKD(1),IBLANK
C CHARACTER*6 PACKED
C DATA IBLANK /1H /
C INITIALIZE NCHMTB
C NCHMTB = 0
C RETURN IF NCHARS IS LE ZERO
C IF (NCHARS.LE.0) RETURN
C SET NC=NUMBER OF CHARS TO BE DECODED
C NC = MIN0 (129,NCHARS)
C READ (PACKED,150) (UNPAKD(I),I=1,NC)
150 FORMAT (129A1)
C CHECK UNPAKD ARRAY AND SET NCHMTB

```

```

C
      DO 200 N = 1,NC
      NN = NC - N + 1
      IF (URPAKD(NN) .NE. IBLANK) GO TO 210
200 CONTINUE
      NN = 0
210 NCHMTB = NN
      RETURN
      END
CVHS12=====
C
      SUBROUTINE VHS12 (MODE,LP,L1,M,U,INCU,UP,C,INCC,ICV,NCV)
C REAL HOUSEHOLDER TRANSFORMATION - COMPUTATION & APPLICATIONS
      MODE,LP,L1,M,INCU,INCC,ICV,NCV
      U(1),UP,C(1)
      REAL
      INTEGER
      DOUBLE PRECISION
      SM,B
      ONE,CL,CLINV,SM1
      FIRST EXECUTABLE STATEMENT
C
      ONE = 1.
C
      IF (0.GE.LP.OR.LP.GE.L1.OR.L1.GT.M) GO TO 9005
      ILP = (LP-1)*INCU+1
      IL1 = (L1-1)*INCU+1
      IM = (M-1)*INCU+1
      CL = ABS(U(ILP))
      IF (MODE.EQ.2) GO TO 15
C
      DO 5 IJ=IL1,IM,INCU
      5 CL = AMAX1(ABS(U(IJ)),CL)
      IF (CL.LE.0.0) GO TO 9005
      CLINV = ONE/CL
      SM = (DBLE(U(ILP))*CLINV)**2
      DO 10 IJ=IL1,IM,INCU
      10 SM = SM+(DBLE(U(IJ))*CLINV)**2
      CONVERT DBLE. PREC. SM TO SNGL.
      PREC. SM1
C
      SM1 = SM
      CL = CL*SORT(SM1)
      IF (U(ILP).GT.0.0) CL = -CL
      UP = U(ILP)-CL
      U(ILP) = CL
      GO TO 20
C
      APPLY THE TRANSFORMATION
      I+U*(U*TI)/B TO C.
C
      15 IF (CL.LE.0.0) GO TO 9005
      20 IF (NCV.LE.0) GO TO 9005
      B = DBLE(UP)*U(ILP)
      B MUST BE NONPOSITIVE HERE. IF B =
      0., RETURN.
C
      IF (B.GE.0.0) GO TO 9005
      B = ONE/B
      I2 = 1-ICV+INCC*(LP-1)
      INCR = INCC*(L1-LP)
      DO 35 J=1,NCV
      12 = I2+ICV
      13 = I2+INCR
      14 = I3
      SM = C(I2)*DBLE(UP)
      DO 25 IJ=IL1,IM,INCU

```

```

      SM = SM+C(I3)*DBLE(U(IJ))
      I3 = I3+INCC
25    CONTINUE
      IF (SM.EQ.0.0) GO TO 35
      SM = SM*B
      C(I2) = C(I2)*SM*DBLE(UP)
      DO 30 IJ=IL1,IM,INCU
        C(I4) = C(I4)+SM*DBLE(U(IJ))
        I4 = I4+INCC
30    CONTINUE
35    CONTINUE
9005 RETURN
      END
-----
      SUBROUTINE SROTG  (SA,SB,SC,SS)
      REAL      SA,SB,SC,SS
      REAL      R,U,V
      SPECIFICATIONS FOR LOCAL VARIABLES
      SPECIFICATIONS FOR LOCAL VARIABLES
      FIRST EXECUTABLE STATEMENT
      IF (ABS(SA).LE.ABS(SB)) GO TO 5
      HERE ABS(SA) .GT. ABS(SB)
      U = SA+SA
      V = SB/U
      NOTE THAT U AND R HAVE THE SIGN OF
      SA
      R = SQRT(.25+V**2)*U
      NOTE THAT SC IS POSITIVE
      SC = SA/R
      SS = V*(SC+SC)
      SB = SS
      SA = R
      RETURN
      HERE ABS(SA) .LE. ABS(SB)
      NOTE THAT U AND R HAVE THE SIGN OF
      SB (R IS IMMEDIATELY STORED IN SA)
      NOTE THAT SS IS POSITIVE
      SS = SB/SA
      SC = V*(SS+SS)
      IF (SC.EQ.0.) GO TO 10
      SB = 1./SC
      RETURN
10    SB = 1.
      RETURN
      HERE SA = SB = 0.
15    SL = 1.
      SS = 0.
      SA = 0.
      SB = 0.
      RETURN
      CSORT=-----
      SUBROUTINE SORT(IAR,NREC,IOUM1,IOUM2)

```

```

C THIS SUBROUTINE SORTS THE INTEGER ARRAY IAR(1) (NREC ELEMENTS LONG)
C IN ASCENDING ORDER
  DIMENSION IAR(1)
  10 ISWAP=0
  DO 50 I=2,NREC
    IF (IAR(I).LT.IAR(I-1)) THEN
      ITEMP=IAR(I)
      IAR(I)=IAR(I-1)
      IAR(I-1)=ITEMP
      ISWAP=1
    END IF
  50 CONTINUE
  IF (ISWAP.EQ.1) GO TO 10
  RETURN
  END

```

Appendix B - GALACTIC Hyperplanes

0	0.9222164	6	27.21381	27.21380	0.0000000E+00	0.0000000E+00
	-5.0040212E-02		0.0000000E+00	0.0000000E+00	0.0000000E+00	28.97421
	0.0000000E+00		0.0000000E+00	0.9446208	6.5553442E-02	-1.519141
	0.0000000E+00		-3.4442861E-03	0.1557674	-1.5272586E-02	
	0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
0	0.8665400	27	24.01917	24.02001	0.0000000E+00	
	-4.9322270E-02		0.0000000E+00	0.0000000E+00	0.0000000E+00	27.4878
	0.0000000E+00		0.9284894	7.2398975E-02	-1.030657	
	0.0000000E+00		-3.7190316E-03	0.1692500	-1.9136427E-02	
	0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
6	27	-0.4023530	-1.049219	0.0000000E+00	0.0000000E+00	
-6.0459729E-03	1.0136234E-04	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.2786539	
0.0000000E+00	0.0000000E+00	-2.6180461E-02	2.1838184E-02	-0.3301796		
0.0000000E+00	0.0000000E+00	-2.8213355E-04	3.7920564E-02	-9.4501488E-03		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
10	6	-0.1857580	-0.1957389	0.0000000E+00	0.0000000E+00	
-3.9836075E-03	3.2360543E-04	0.0000000E+00	0.0000000E+00	-0.2728751		
0.0000000E+00	0.0000000E+00	3.9934047E-02	-3.3462569E-05	0.2180881		
0.0000000E+00	0.0000000E+00	-2.6336135E-04	-1.8169048E-04	-5.1708411E-02		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
10	27	0.1097895	0.1098012	0.0000000E+00	0.0000000E+00	
2.1876304E-03	1.2442684E-04	0.0000000E+00	0.0000000E+00	7.6426141E-02		
0.0000000E+00	0.0000000E+00	-8.9947833E-03	8.7648146E-03	0.6156864		
0.0000000E+00	0.0000000E+00	-5.4559211E-04	-7.7063115E-03	-6.6956277E-03		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
1	6	-0.1423384	-0.1493975	0.0000000E+00	0.0000000E+00	
1.4435121E-02	-7.2491996E-04	0.0000000E+00	0.0000000E+00	2.0684099E-02		
0.0000000E+00	0.0000000E+00	-3.9675612E-02	-5.8753323E-03	0.1195173		
0.0000000E+00	0.0000000E+00	-2.0958172E-05	1.5517431E-02	3.4307712E-03		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
1	27	-2.283033	-2.288582	0.0000000E+00	0.0000000E+00	
-3.3627453E-03	-2.3712330E-03	0.0000000E+00	0.0000000E+00	3.1622738E-02		
0.0000000E+00	0.0000000E+00	1.8726872E-02	-7.1243709E-04	4.0088929E-02		
0.0000000E+00	0.0000000E+00	-1.2975630E-04	-5.4342352E-04	-3.1034152E-03		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
21	6	0.1572241	7.8537047E-02	0.0000000E+00	0.0000000E+00	
7.5499224E-03	-8.3128945E-04	0.0000000E+00	0.0000000E+00	0.4442599		
0.0000000E+00	0.0000000E+00	-2.5632134E-02	-1.3788607E-02	-3.7654266E-02		
0.0000000E+00	0.0000000E+00	-9.6157390E-05	6.7412689E-02	6.7152698E-03		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
21	27	-2.949393	-2.952818	0.0000000E+00	0.0000000E+00	
-1.5602403E-02	-1.0362453E-03	0.0000000E+00	0.0000000E+00	-0.7598284		
0.0000000E+00	0.0000000E+00	1.0475507E-02	-5.3582452E-03	2.3897994E-02		
0.0000000E+00	0.0000000E+00	-4.0323175E-05	-2.7148691E-03	3.1215919E-03		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
20	6	-1.923258	-1.982673	0.0000000E+00	0.0000000E+00	
-4.2127953E-03	-1.8211971E-03	0.0000000E+00	0.0000000E+00	-0.1661534		
0.0000000E+00	0.0000000E+00	2.9783852E-02	6.6771149E-04	0.1847285		
0.0000000E+00	0.0000000E+00	-1.6040249E-05	-4.2282321E-02	-1.0249301E-02		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
20	27	-1.924175	-2.037607	0.0000000E+00	0.0000000E+00	
-5.1516392E-03	-1.8020751E-03	0.0000000E+00	0.0000000E+00	-0.1524218		
0.0000000E+00	0.0000000E+00	2.9563380E-02	9.4146974E-04	0.1948719		
0.0000000E+00	0.0000000E+00	-7.6944243E-06	-4.1933142E-02	-1.0111452E-02		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		
7	22	0.5795203	0.5227754	0.0000000E+00	0.0000000E+00	
3.1011046E-03	-5.0478679E-04	0.0000000E+00	0.0000000E+00	0.3108462		
0.0000000E+00	0.0000000E+00	-2.3448415E-02	3.0095249E-02	0.2768602		
0.0000000E+00	0.0000000E+00	9.8913180E-05	1.6721459E-02	5.9865196E-03		
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00		

6.3395808E-03	-8.1227126E-04	0.0000000E+00	-1.061939	0.0000000E+00
0.0000000E+00	0.0000000E+00	-4.7657855E-02	0.0000000E+00	-0.1962710
0.0000000E+00	0.0000000E+00	-1.3219585E-06	9.0055102E-03	1.6853672E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	1.2956944E-03
2.1493365E-03	-5.8429094E-05	0.0000000E+00	-1.286273	0.0000000E+00
0.0000000E+00	0.0000000E+00	-1.6388714E-02	0.0000000E+00	-0.3912630
0.0000000E+00	0.0000000E+00	-8.9570880E-05	-3.3054404E-02	-0.2706596
0.0000000E+00	0.0000000E+00	0.0000000E+00	-1.0711649E-02	-4.0136306E-03
2.2720976E-02	4.4348044E-04	0.0000000E+00	-1.468430	0.0000000E+00
0.0000000E+00	0.0000000E+00	-8.1784735E-03	0.0000000E+00	-2.7344901E-02
0.0000000E+00	0.0000000E+00	-7.1745795E-05	5.3002522E-04	-2.4864692E-04
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
-2.7961617E-03	-1.4191014E-03	0.0000000E+00	-2.518939	0.0000000E+00
0.0000000E+00	0.0000000E+00	1.4517985E-02	6.9554118E-03	0.1378282
0.0000000E+00	0.0000000E+00	3.4872988E-05	1.8711749E-02	6.7446232E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
2.0769412E-02	-9.6607284E-04	0.0000000E+00	3.8033158E-02	0.0000000E+00
0.0000000E+00	0.0000000E+00	1.7119616E-02	0.0000000E+00	-4.6494271E-04
0.0000000E+00	0.0000000E+00	-8.3054110E-05	-6.6154227E-03	-2.1943608E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
1.1542683E-03	-1.7420055E-03	0.0000000E+00	-0.617015	0.0000000E+00
0.0000000E+00	0.0000000E+00	6.9772433E-03	0.0000000E+00	0.2514621
0.0000000E+00	0.0000000E+00	2.152532E-04	-1.1804328E-02	0.4807620
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	8.1892591E-03
-2.0064194E-02	-4.6435519E-04	0.0000000E+00	-0.9318274	0.0000000E+00
0.0000000E+00	0.0000000E+00	-3.1670113E-03	0.0000000E+00	0.5680899
0.0000000E+00	0.0000000E+00	2.3319873E-05	-8.7503735E-03	6.3020691E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	1.1319662E-02	1.9562445E-03
-3.7568253E-03	4.1548777E-04	0.0000000E+00	-1.693602	0.0000000E+00
0.0000000E+00	0.0000000E+00	1.3587160E-02	0.0000000E+00	-0.7813047
0.0000000E+00	0.0000000E+00	-2.6481459E-04	1.3430241E-02	-0.4653442
0.0000000E+00	0.0000000E+00	0.0000000E+00	-2.3191137E-02	-5.4212278E-03
-2.0064194E-02	-4.6435519E-04	0.0000000E+00	-0.5941176	0.0000000E+00
0.0000000E+00	0.0000000E+00	-3.1670113E-03	0.0000000E+00	0.5680899
0.0000000E+00	0.0000000E+00	2.3319873E-05	-8.7503735E-03	6.3020691E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	1.1319662E-02	1.9562445E-03
-2.0064194E-02	-4.6435519E-04	0.0000000E+00	-0.9318274	0.0000000E+00
0.0000000E+00	0.0000000E+00	-3.1670113E-03	0.0000000E+00	0.5680899
0.0000000E+00	0.0000000E+00	2.3319873E-05	-8.7503735E-03	6.3020691E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	1.1319662E-02	1.9562445E-03
-1.0336907E-02	-1.0526507E-03	0.0000000E+00	-2.682809	0.0000000E+00
0.0000000E+00	0.0000000E+00	8.1318943E-03	0.0000000E+00	-0.9193538
0.0000000E+00	0.0000000E+00	6.4787317E-05	7.5071254E-03	0.1464850
0.0000000E+00	0.0000000E+00	0.0000000E+00	2.0933010E-02	7.4759726E-03
-4.9271611E-03	-1.6403894E-04	0.0000000E+00	-2.105929	0.0000000E+00
0.0000000E+00	0.0000000E+00	-2.7247025E-03	0.0000000E+00	-1.121221
0.0000000E+00	0.0000000E+00	6.2482643E-05	1.4703546E-02	8.0510251E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	1.4262024E-02	5.8542574E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00

27	23	0.9318274	0.7368441	0.0000000E+00
2.0064194E-02	4.6435519E-04	0.0000000E+00	0.0000000E+00	0.0000000E+00
0.0000000E+00	0.0000000E+00	3.1670113E-03	8.7503735E-03	-6.3020691E-02
0.0000000E+00	0.0000000E+00	-2.3319873E-05	-1.1319662E-02	-1.9562445E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	3	-0.2197392	-0.2180968	0.0000000E+00
7.7350932E-04	-2.7156188E-04	0.0000000E+00	0.0000000E+00	-1.1509622E-02
0.0000000E+00	0.0000000E+00	-3.6839896E-03	-1.4136495E-03	-0.4094222
0.0000000E+00	0.0000000E+00	3.9803408E-04	1.5038691E-02	-7.6856986E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	4	-0.9735251	-0.9924156	0.0000000E+00
-2.0114245E-05	8.2086292E-05	0.0000000E+00	0.0000000E+00	-1.170877
0.0000000E+00	0.0000000E+00	3.7385977E-03	8.0561414E-03	0.1476528
0.0000000E+00	0.0000000E+00	1.6620012E-04	1.9310748E-02	8.2446365E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	6	-2.097861	-2.150858	0.0000000E+00
-5.9131165E-03	-1.7365658E-03	0.0000000E+00	0.0000000E+00	-0.5039683
0.0000000E+00	0.0000000E+00	2.4914838E-02	8.5098054E-03	0.2365472
0.0000000E+00	0.0000000E+00	1.0436781E-04	8.9248661E-03	4.4868723E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	27	-0.4254984	-0.4992822	0.0000000E+00
-3.0243504E-04	9.5086062E-04	0.0000000E+00	0.0000000E+00	-0.1760031
0.0000000E+00	0.0000000E+00	-1.9096304E-02	2.0192651E-02	-0.3167804
0.0000000E+00	0.0000000E+00	-2.2781771E-04	-5.9769765E-02	-1.3833243E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
3	4	-1.060428	-1.079199	0.0000000E+00
8.2350819E-04	5.1913561E-05	0.0000000E+00	0.0000000E+00	-1.190682
0.0000000E+00	0.0000000E+00	6.1223372E-03	8.9555904E-03	8.8194147E-02
0.0000000E+00	0.0000000E+00	1.2293969E-04	1.5393170E-02	7.0810951E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
3	6	-1.887722	-1.998381	0.0000000E+00
6.7755260E-04	-1.8996334E-03	0.0000000E+00	0.0000000E+00	-0.4732971
0.0000000E+00	0.0000000E+00	2.6933979E-02	7.2805346E-03	0.2019476
0.0000000E+00	0.0000000E+00	6.3447304E-05	7.2106305E-03	3.6148161E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
3	27	-2.558975	-2.602511	0.0000000E+00
-5.1078871E-03	-2.2398222E-03	0.0000000E+00	0.0000000E+00	3.0157549E-02
0.0000000E+00	0.0000000E+00	1.6275687E-02	5.2219946E-03	-0.1239222
0.0000000E+00	0.0000000E+00	-2.3824138E-04	-8.3891163E-03	-4.0908526E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
4	6	1.113510	1.109852	0.0000000E+00
1.4540923E-03	-2.7347438E-04	0.0000000E+00	0.0000000E+00	1.103516
0.0000000E+00	0.0000000E+00	-7.3536062E-03	-1.2401587E-03	-0.2022413
0.0000000E+00	0.0000000E+00	-1.8350588E-04	3.9429858E-02	2.6987006E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
4	27	-0.3699014	-0.6129559	0.0000000E+00
-5.2240539E-05	-4.3425604E-04	0.0000000E+00	0.0000000E+00	0.6460940
0.0000000E+00	0.0000000E+00	-1.1936553E-02	1.6261997E-02	-0.4271832
0.0000000E+00	0.0000000E+00	-3.9737124E-04	-2.0365780E-02	-1.0088839E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
21	2	-0.3935746	-0.3935728	0.0000000E+00
2.7480145E-04	-5.0791539E-04	0.0000000E+00	0.0000000E+00	1.2662138E-02
0.0000000E+00	0.0000000E+00	-3.9376500E-03	-4.5612259E-03	-0.6389677
0.0000000E+00	0.0000000E+00	5.1584718E-04	-3.9245598E-03	2.5574410E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
21	3	0.1310589	0.1310690	0.0000000E+00
2.3520799E-04	2.9619815E-04	0.0000000E+00	0.0000000E+00	9.3356380E-03
0.0000000E+00	0.0000000E+00	-3.5944888E-03	-4.0897918E-03	0.6267993
0.0000000E+00	0.0000000E+00	-5.5186014E-04	-6.7999749E-03	2.1235060E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00

21	4	0.1385698	0.1385701	0.0000000E+00
9.1947501E-04	1.1418112E-04	0.0000000E+00	0.0000000E+00	4.042344E-02
0.0000000E+00	0.0000000E+00	-7.8544803E-03	-5.0654933E-03	0.6536500
0.0000000E+00	0.0000000E+00	-4.8479735E-04	-5.3164833E-03	3.3964787E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
20	7	0.1284901	0.1285026	0.0000000E+00
-6.4352131E-03	4.2266780E-04	0.0000000E+00	0.0000000E+00	7.0380233E-02
0.0000000E+00	0.0000000E+00	2.0308526E-02	-2.6275124E-02	-0.4408797
0.0000000E+00	0.0000000E+00	2.4214239E-04	6.5260483E-03	8.2986029E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
20	22	0.3617558	0.3415487	0.0000000E+00
-8.8951e32E-04	1.1726847E-04	0.0000000E+00	0.0000000E+00	-6.4811178E-02
0.0000000E+00	0.0000000E+00	1.3296803E-02	-1.3457318E-02	0.1466104
0.0000000E+00	0.0000000E+00	2.9155856E-04	8.3990218E-03	-8.5888989E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
20	5	-9.2567496E-02	-0.1034234	0.0000000E+00
1.7773038E-02	-9.1011351E-04	0.0000000E+00	0.0000000E+00	-0.2378222
0.0000000E+00	0.0000000E+00	2.7512822E-02	-2.0900595E-03	0.1544586
0.0000000E+00	0.0000000E+00	4.3214222E-05	-1.5703907E-02	-3.0972422E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
20	23	-0.4754973	-0.5475546	0.0000000E+00
-4.3455209E-03	-9.1509428E-04	0.0000000E+00	0.0000000E+00	-0.2918059
0.0000000E+00	0.0000000E+00	1.3986363E-02	-1.1788827E-02	0.5721745
0.0000000E+00	0.0000000E+00	3.8312699E-04	2.0491051E-02	8.2345614E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
21	7	0.4057466	0.3755752	0.0000000E+00
-8.5744858E-03	7.7419833E-04	0.0000000E+00	0.0000000E+00	6.7189477E-02
0.0000000E+00	0.0000000E+00	3.9615002E-02	-1.2663011E-02	-0.2420823
0.0000000E+00	0.0000000E+00	-1.6314992E-04	2.9463375E-02	8.2678217E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
21	22	0.2068170	0.2092525	0.0000000E+00
2.6256937E-05	2.7108443E-04	0.0000000E+00	0.0000000E+00	2.6222460E-03
0.0000000E+00	0.0000000E+00	-5.5122253E-04	-4.0405453E-04	0.6905332
0.0000000E+00	0.0000000E+00	-5.3170341E-04	-1.3061181E-03	4.4055162E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
21	5	0.4055288	0.1361879	0.0000000E+00
4.2272680E-03	6.6679218E-05	0.0000000E+00	0.0000000E+00	0.1047683
0.0000000E+00	0.0000000E+00	2.4118461E-03	-3.8150143E-02	3.8006034E-02
0.0000000E+00	0.0000000E+00	1.4544026E-05	1.4596310E-02	3.8561109E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
21	23	-2.255741	-2.279994	0.0000000E+00
-2.6099144E-03	-1.2779168E-03	0.0000000E+00	0.0000000E+00	-0.9360097
0.0000000E+00	0.0000000E+00	1.3183454E-02	3.2598197E-03	0.2349887
0.0000000E+00	0.0000000E+00	1.1475040E-04	2.6104683E-02	9.1243815E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
1	7	-2.3550333E-02	-1.9627597E-02	0.0000000E+00
-1.5966163E-04	1.0430538E-05	0.0000000E+00	0.0000000E+00	-1.4075175E-02
0.0000000E+00	0.0000000E+00	1.2034351E-03	-1.2932327E-03	-7.6851532E-02
0.0000000E+00	0.0000000E+00	9.8525787E-05	2.7641548E-02	-0.1019245
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
1	22	0.3280837	0.3276035	0.0000000E+00
1.2556893E-04	2.9364397E-04	0.0000000E+00	0.0000000E+00	-1.0370051E-02
0.0000000E+00	0.0000000E+00	5.6057787E-03	5.8399723E-03	0.5604030
0.0000000E+00	0.0000000E+00	-5.1407632E-04	3.2191694E-02	-3.4508687E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
1	5	-0.1387606	-0.1387615	0.0000000E+00
1.9325523E-02	-1.1047518E-03	0.0000000E+00	0.0000000E+00	8.4972270E-03
0.0000000E+00	0.0000000E+00	-3.9068935E-03	-2.4464075E-03	-0.2459702
0.0000000E+00	0.0000000E+00	2.2022489E-04	-9.8375638E-04	-1.2694754E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	

-3.8232193E-03	1	23	0.2757462	0.2017785	0.0000000E+00
0.0000000E+00		2.2681162E-04	0.0000000E+00	0.0000000E+00	-0.2748562
0.0000000E+00		0.0000000E+00	3.8409300E-02	2.5079228E-02	0.1491033
0.0000000E+00		0.0000000E+00	1.0512739E-04	1.3133987E-03	9.2581956E-04
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
-1.8727807E-03	10	7	-0.2332405	-0.2061299	0.0000000E+00
0.0000000E+00		4.8079339E-04	0.0000000E+00	0.0000000E+00	-9.1713540E-02
0.0000000E+00		0.0000000E+00	5.8340430E-03	1.6465910E-02	0.4578562
0.0000000E+00		0.0000000E+00	-2.7490119E-04	7.2498590E-02	-2.1147290E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
4.8990099E-04	10	22	0.1029203	0.1109052	0.0000000E+00
0.0000000E+00		1.9362812E-04	0.0000000E+00	0.0000000E+00	-1.1586168E-02
0.0000000E+00		0.0000000E+00	-3.4052713E-03	1.0563395E-03	0.6447840
0.0000000E+00		0.0000000E+00	-5.4064224E-04	4.1272528E-03	2.4662258E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
0.5842420	10	5	-1.975560	-1.975577	0.0000000E+00
0.0000000E+00		-1.8510444E-02	0.0000000E+00	0.0000000E+00	-8.293071
0.0000000E+00		0.0000000E+00	0.6258693	0.1231744	-0.8652425
0.0000000E+00		0.0000000E+00	-1.5126106E-03	-0.6009062	2.351106
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
-1.4984467E-03	10	23	0.2736165	0.5548023	0.0000000E+00
0.0000000E+00		3.5825159E-04	0.0000000E+00	0.0000000E+00	0.2030284
0.0000000E+00		0.0000000E+00	3.2119798E-03	1.3291849E-03	0.6619402
0.0000000E+00		0.0000000E+00	-4.4525723E-04	4.3248504E-02	8.4720310E-03
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
-1.1082066E-03	0	7	0.8240990	0.9276105	0.0000000E+00
0.0000000E+00		1.8902248E-04	0.0000000E+00	0.0000000E+00	0.2249505
0.0000000E+00		0.0000000E+00	2.0155622E-02	1.9414650E-02	0.2750238
0.0000000E+00		0.0000000E+00	4.4288008E-05	4.0131032E-02	6.5958038E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
0.7826716	0	22	21.73051	21.73053	0.0000000E+00
0.0000000E+00		-4.3509316E-02	0.0000000E+00	0.0000000E+00	23.32464
0.0000000E+00		0.0000000E+00	0.8472973	0.1072113	-0.1456653
0.0000000E+00		0.0000000E+00	-3.6448685E-03	0.1075505	-3.3292517E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
1.021026	0	5	28.22051	28.22050	0.0000000E+00
0.0000000E+00		-5.6861322E-02	0.0000000E+00	0.0000000E+00	30.42640
0.0000000E+00		0.0000000E+00	1.114380	0.1269589	-0.3560690
0.0000000E+00		0.0000000E+00	-4.6405080E-03	0.1440598	-4.0388469E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
-2.4207475E-05	0	23	0.2599925	0.2599437	0.0000000E+00
0.0000000E+00		2.7965786E-04	0.0000000E+00	0.0000000E+00	3.4267720E-02
0.0000000E+00		0.0000000E+00	2.5798209E-04	3.3248086E-03	0.5580868
0.0000000E+00		0.0000000E+00	-4.8970216E-04	1.1635082E-02	5.1717546E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
2.3632792E-03	2	20	-0.5007082	-0.5007089	0.0000000E+00
0.0000000E+00		-6.218681E-04	0.0000000E+00	0.0000000E+00	-5.1193018E-02
0.0000000E+00		0.0000000E+00	-1.1535180E-02	1.3952782E-02	-0.5413191
0.0000000E+00		0.0000000E+00	4.6965815E-04	1.7259564E-02	2.5915673E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
-5.7185930E-04	3	20	-5.0488465E-02	-5.0487719E-02	0.0000000E+00
0.0000000E+00		-1.4394653E-04	0.0000000E+00	0.0000000E+00	2.7651181E-03
0.0000000E+00		0.0000000E+00	4.5211944E-03	5.2238707E-03	-0.3373621
0.0000000E+00		0.0000000E+00	2.6006749E-04	3.4267720E-02	-8.5785717E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	
-3.2799810E-04	20	4	1.2975411E-02	1.2974638E-02	0.0000000E+00
0.0000000E+00		-7.6797725E-05	0.0000000E+00	0.0000000E+00	-4.3699462E-03
0.0000000E+00		0.0000000E+00	4.8961971E-05	4.7011888E-03	-4.8800491E-02
0.0000000E+00		0.0000000E+00	1.6622635E-04	-3.9245475E-02	9.6609071E-02
0.0000000E+00		0.0000000E+00	0.0000000E+00	0.0000000E+00	

1	2	0.511591	0.5483046	0.0000000E+00
-6.8386371E-04	4.0195975E-04	0.0000000E+00	0.0000000E+00	3.9590657E-02
0.0000000E+00	7.2772149E-03	7.3158983E-03	0.5746222	
0.0000000E+00	-4.2677630E-04	2.5514092E-02	-4.9216140E-02	
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
1	3	0.2963204	0.2965649	0.0000000E+00
-6.4314139E-04	3.6349706E-04	0.0000000E+00	0.0000000E+00	-6.6771987E-04
0.0000000E+00	0.0000000E+00	4.4955690E-03	1.4898469E-03	0.4907922
0.0000000E+00	0.0000000E+00	-3.7963822E-04	2.2022523E-02	-7.0144102E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
1	4	0.3013018	0.3013037	0.0000000E+00
-7.7932555E-04	3.8106649E-04	0.0000000E+00	0.0000000E+00	-1.7103240E-03
0.0000000E+00	0.0000000E+00	6.0579227E-03	6.0029863E-04	0.6357226
0.0000000E+00	0.0000000E+00	-5.4787961E-04	1.5726036E-03	1.8731272E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
2	10	-0.1649746	-0.1634048	0.0000000E+00
-5.5235112E-05	-3.1786819E-04	0.0000000E+00	0.0000000E+00	-1.0655276E-02
0.0000000E+00	0.0000000E+00	5.8669591E-04	1.7095235E-03	-0.6418737
0.0000000E+00	0.0000000E+00	5.7006121E-04	1.0922260E-03	5.6725909E-04
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
3	10	0.2472814	0.2473280	0.0000000E+00
-1.5656333E-03	-1.7208478E-04	0.0000000E+00	0.0000000E+00	6.7100063E-02
0.0000000E+00	0.0000000E+00	9.6575525E-03	1.4601735E-02	-0.5838240
0.0000000E+00	0.0000000E+00	4.9647613E-04	2.4369385E-02	4.4312496E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
10	4	0.3886607	0.4994883	0.0000000E+00
-1.1849942E-03	2.9601471E-04	0.0000000E+00	0.0000000E+00	0.3216445
0.0000000E+00	0.0000000E+00	1.4153545E-03	1.4043109E-03	0.6345115
0.0000000E+00	0.0000000E+00	-2.8472478E-04	-6.8457223E-02	8.4627448E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
0	2	0.1542452	0.2216018	0.0000000E+00
4.6465229E-03	-2.9410623E-04	0.0000000E+00	0.0000000E+00	4.6789087E-02
0.0000000E+00	0.0000000E+00	-2.1083590E-02	1.2364086E-02	0.4006078
0.0000000E+00	0.0000000E+00	-1.1092038E-04	3.1567797E-02	-7.0265412E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
0	3	0.1211064	0.2582814	0.0000000E+00
2.4283908E-03	-1.5722604E-04	0.0000000E+00	0.0000000E+00	1.3884456E-02
0.0000000E+00	0.0000000E+00	-1.2961977E-02	1.4028946E-02	0.3762399
0.0000000E+00	0.0000000E+00	-1.2358476E-04	3.2897972E-02	-7.8430772E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
0	4	0.1950551	0.1994090	0.0000000E+00
-4.8026396E-04	3.1156171E-04	0.0000000E+00	0.0000000E+00	-5.1489808E-03
0.0000000E+00	0.0000000E+00	3.1105785E-03	1.0997916E-04	0.6633888
0.0000000E+00	0.0000000E+00	-5.4658845E-04	-4.5761033E-03	9.8880753E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
21	1	-0.1713047	-0.1713012	0.0000000E+00
-2.0414009E-03	-1.8875749E-04	0.0000000E+00	0.0000000E+00	5.1234782E-02
0.0000000E+00	0.0000000E+00	7.7920733E-03	-8.8252081E-03	-0.5322345
0.0000000E+00	0.0000000E+00	3.0975265E-04	-7.9713175E-03	7.0511021E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
21	20	-7.2240822E-02	-4.2462621E-02	0.0000000E+00
-1.9702390E-03	-1.1756549E-04	0.0000000E+00	0.0000000E+00	6.5879524E-02
0.0000000E+00	0.0000000E+00	1.1132965E-02	-3.2998407E-03	-0.7314168
0.0000000E+00	0.0000000E+00	4.5274838E-04	1.4669186E-02	-1.3945083E-04
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
1	20	0.2794988	0.2794958	0.0000000E+00
4.0013026E-03	4.1886688E-05	0.0000000E+00	0.0000000E+00	-1.2021876E-02
0.0000000E+00	0.0000000E+00	-1.0945338E-02	1.0942113E-02	0.6504494
0.0000000E+00	0.0000000E+00	-4.2033434E-04	1.1498610E-02	-3.1931791E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	

21	10	-0.4329941	-0.2422430	0.0000000E+00
2	3396760E-03	-3.8041692E-04	0.0000000E+00	0.0000000E+00
0	0.0000000E+00	-7.3157344E-03	-1.4265542E-03	-0.1273370
0	0.0000000E+00	4.8975350E-04	2.7889628E-02	-2.7262237E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
20	10	-0.4110589	-0.1805754	0.0000000E+00
3	4529307E-03	-3.0863678E-04	0.0000000E+00	-0.2077786
0	0.0000000E+00	-8.2077282E-03	-2.1495740E-03	-0.6030180
0	0.0000000E+00	4.5786559E-04	4.0426899E-02	-2.5238711E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0	20	23.07658	24.38633	0.0000000E+00
1	235538	-5.5918526E-02	0.0000000E+00	42.03928
0	0.0000000E+00	0.5945719	0.504909	-13.15489
0	0.0000000E+00	-1.3037957E-02	-0.6719638	-0.2679172
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
21	0	-20.16562	-20.16585	0.0000000E+00
-1	397251	6.9062322E-02	0.0000000E+00	-42.86494
0	0.0000000E+00	-0.7038303	-0.9178057	12.28921
0	0.0000000E+00	1.3084883E-02	1.110618	0.3762960
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
1	10	0.4961410	0.5280035	0.0000000E+00
-1	8672388E-04	5.2940213E-05	0.0000000E+00	0.1234468
0	0.0000000E+00	7.4593048E-03	1.1848936E-02	-1.3010859E-02
0	0.0000000E+00	1.2015631E-04	4.2882677E-02	-9.0552568E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
1	0	0.2336065	0.2568044	0.0000000E+00
-1	4813519E-03	3.835839E-04	0.0000000E+00	-1.1672183E-02
0	0.0000000E+00	7.0010051E-03	-7.3218596E-04	0.3321991
0	0.0000000E+00	-2.7845841E-04	2.6801020E-02	-8.6197518E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0	0	-0.2321022	2.5051221E-02	0.0000000E+00
3	9048633E-03	4.241768E-04	0.0000000E+00	-8.2190022E-02
0	0.0000000E+00	-8.3781667E-03	4.4831430E-04	-0.4609540
0	0.0000000E+00	2.9259894E-04	6.3622579E-02	-5.5368930E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	7	7.3668428E-02	7.3668830E-02	0.0000000E+00
-1	7890703E-03	3.3631400E-04	0.0000000E+00	-6.2551722E-02
0	0.0000000E+00	1.0815877E-02	-5.4139541E-03	0.1017401
0	0.0000000E+00	-8.5408581E-05	1.9421577E-02	-9.8950692E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
2	22	0.3944690	0.4119532	0.0000000E+00
1	5843599E-03	1.0278248E-04	0.0000000E+00	1.7171111E-02
0	0.0000000E+00	1.8757958E-02	6.3992417E-03	0.3582918
0	0.0000000E+00	-3.4819610E-04	-5.3887633E-03	7.6162145E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
5	0.8188578	0.7408233	0.0000000E+00	0.0000000E+00
-2	3480231E-04	0.0000000E+00	0.0000000E+00	-0.1359558
0	0.0000000E+00	-1.1981551E-02	-7.8201024E-03	0.6144885
0	0.0000000E+00	4.2819351E-04	3.9935797E-02	1.2465645E-02
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
23	-1.998477	-2.335875	0.0000000E+00	0.0000000E+00
-2	0845931E-03	-1.2637860E-03	0.0000000E+00	-0.9638517
0	0.0000000E+00	1.1659215E-02	6.1305990E-03	0.2097973
0	0.0000000E+00	9.9564073E-05	2.3369001E-02	8.3339429E-03
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
7	0.1658641	0.1524081	0.0000000E+00	0.0000000E+00
-3	6818380E-03	9.5083250E-04	0.0000000E+00	-0.1904521
0	0.0000000E+00	3.2595232E-02	-2.4979820E-02	-0.2175574
0	0.0000000E+00	-4.4529232E-05	-1.6139196E-02	-5.1972037E-03
0	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00

3	22	-0.1314286	-0.1528783	0.0000000E+00
7.7515920E-03	-9.7303139E-04	0.0000000E+00	0.0000000E+00	5.5731092E-02
0.0000000E+00	0.0000000E+00	-3.5994649E-02	1.8355209E-02	0.2615914
0.0000000E+00	0.0000000E+00	9.3401897E-05	1.9759661E-02	6.1906716E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
3	5	-2.148884	-2.170104	0.0000000E+00
-1.3511101E-04	-1.5010128E-03	0.0000000E+00	0.0000000E+00	-0.9038910
0.0000000E+00	0.0000000E+00	1.6256787E-02	6.5549118E-03	0.1306692
0.0000000E+00	0.0000000E+00	2.3591552E-05	1.7382551E-02	6.2891878E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
3	23	-1.859904	-2.158417	0.0000000E+00
1.4576917E-03	-1.3715532E-03	0.0000000E+00	0.0000000E+00	-0.9311829
0.0000000E+00	0.0000000E+00	1.3946485E-02	5.5924566E-03	0.2010224
0.0000000E+00	0.0000000E+00	8.5209685E-05	2.1653350E-02	7.7461298E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
7	4	0.6376910	0.6375058	0.0000000E+00
1.3573484E-04	1.8903003E-04	0.0000000E+00	0.0000000E+00	0.3010593
0.0000000E+00	0.0000000E+00	-2.0278781E-04	3.8731347E-03	0.4360211
0.0000000E+00	0.0000000E+00	2.2137816E-04	-9.4285466E-02	-8.8974442E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
22	4	0.6798738	0.6720510	0.0000000E+00
-4.2617293E-03	5.7044980E-04	0.0000000E+00	0.0000000E+00	0.1892032
0.0000000E+00	0.0000000E+00	3.0309694E-02	-1.4207790E-02	0.2523862
0.0000000E+00	0.0000000E+00	1.6575288E-04	-7.0536904E-02	-8.1682298E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
4	5	1.861675	1.858364	0.0000000E+00
1.9169506E-02	1.4478797E-03	0.0000000E+00	0.0000000E+00	-0.1949757
0.0000000E+00	0.0000000E+00	-1.6165713E-03	2.2439579E-04	-9.4212934E-02
0.0000000E+00	0.0000000E+00	5.7865065E-05	-1.0996808E-02	-1.9000837E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
4	23	-2.263177	-2.450073	0.0000000E+00
1.3261826E-03	-1.4671807E-03	0.0000000E+00	0.0000000E+00	-0.9167992
0.0000000E+00	0.0000000E+00	1.5509252E-02	6.7464635E-03	0.1341186
0.0000000E+00	0.0000000E+00	2.8684351E-05	1.8010058E-02	6.5048225E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	

Appendix C - EFAGHY Listing

[illegible]

TTTTT	EEEE	N	CCCC	EEEE
T	E	N	C	E
T	E	N	C	D
T	E	NN	C	D
T	EEEE	N	C	D
T	E	N	C	D
T	E	N	C	D
T	E	N	C	D
T	EEEE	N	CCCC	EEEE

[illegible][illegible]

File_\$IDUAD0:[EFBLBN.FOR;216 (5654,282.0), last revised on 20-JUL-1987 17:12, is a 1'2 block sequential file owned by UIC [DC01,DECNET]. The records are variable length with implied (CR) carriage control. The longest record is 72 bytes.

JOB EFBLBN (432) queued to TXA15 on 20-JUL-1987 17:13 by user DECNET, UIC 10E01, DECNET11, under account 9903 at priority 100.
started on printer AEE251\$TXA15: on 20-JUL-1987 17:56 from queue TXA15.

[illegible]

```

*5 NINF
PARAMETER (NVMX=14)
PARAMETER (INMX=20, ISETMAX=14)
PARAMETER (INMX4=INMX*4, ISETMAX4=ISETMAX*4, NPLANMX=10)
PARAMETER (NSPAIR=(ISETMAX*(ISETMAX-1))/2, ICH=11)
PARAMETER (NFILMAX=15, ISTM12=2*ISETMAX, ISETMAX2=ISETMAX*2)
DIMENSION TEMP(NPAIR, INMX4), ITEMP(NPAIR, INMX4)
8. TEMPV(NSPAIR), BUFFER(INMX), IVOTE(ISETMAX, 6), PERTURB(INMX)
8. ISETID(ISETMAX), IDSETI(100), IBUFFER(INMX), NPINSET(ISETMAX)
8. ISETCOM(ISTMX12), ISETOUT(ISETMAX), VOTE(ISETMAX, 5)
8. ISCOREP(NSPAIR, 8), IPLANE(NPLANMX), ISCORES(ISETMAX, 5)
8. ISCORES(ISETMAX, ISETMAX2, 5), IW1(NSPAIR, 3), W1(NSPAIR, 3)
8. WSET(ISETMAX), ISCOREBP(14), IW2(NSPAIR, 3), W2(NSPAIR, 3)
8. IDCHOICE(ISETMAX, ICH, ISETMAX), XDCHOICE(ISETMAX, ICH, ISETMAX)
8. IHPT(20)
EQUIVALENCE (BUFFER, IBUFFER), (ITEMP, TEMPM), (W1, IW1), (W2, IW2)
8. (IDCHOICE, XDCHOICE)
C. CHARACTER *8 DATE1
CHARACTER *64 FILIN(NFILMAX), FILEHO, FILEVOTE, FILECR
CHARACTER *1CH1, CH2, CH3, CH4, CH5, CHASK, CHBL1, CHV, CHCF
8. CHGT, CHEQ, CHLT, CHNO, CHRB, CHLB, CHB1, CHB2
CHARACTER *3CHYN, C*NOT
CHARACTER *7CHSETCL, CHSET, CHCLS, CHMODE, CHBAT, CHTSH, CHBL
CHARACTER *18REMCF
LOGICAL EXIS
INTEGER H
REAL LHS1, LHS?
ISTOP=0
IBITS=2**30-1
CHASK = ' '
CHV = 'Y'
CHBL1 =
CHBL2 = 'I'
CHLB = 'I'
CHNOT = 'NOT'
CHBL = ' '
CHSET = 'SET'
CHCLS = 'CLUSTER'
CHBAT = 'BATCH'
CHTSH = 'TSHARE'
FILECR = ' '
ICASE=0
VBIGNO=1.E30
60 FORMAT(15I4)
70 FORMAT(10F7.3)
71 FORMAT(15, 16, 10X, A1, F10.5, 10X, F10.5, A1, F10.5, 16)
72 FORMAT(15, 16, 10X, A1, F10.5, F10.5, F10.5, A1, 10X, 16)
73 FORMAT(15, 16, F10.5, A1, F10.5, 10X, F10.5, A1, 10X, 16)
80 FORMAT(15, 216, 15, A1, 14, 315, A1, F4.1, F5.1, F6.1, A1, F5.1, A1, F5.1, A1)
81 FORMAT(15, 216, 15, A1, 14, 315, A1, F4.1, F5.1, F6.1, A1, F5.1, A1, F5.1, A1)
82 FORMAT(15, 16, 15, A1, 14, 315, A1, F4.1, F5.1, F6.1, A1, F5.1, A1, F5.1, A1)
95 FORMAT(17, 1414, F8.3)
96 FORMAT(17, 1414, A8)
85 FORMAT(13, 14, 15, 416, F6.0, 416, F6.0)
86 FORMAT(13, 14, 15, 416, F6.0, A30)
87 FORMAT(13, 14, 15, 30X, 416, F6.0)
91 FORMAT(16, 16, 1314)
92 FORMAT(16, 16, 1314)
93 FORMAT(12, F5.2, 13F5.2)
94 FORMAT(13, 14, 1315)

```

```

88 FORMAT(A18,519)
89 FORMAT(15,A13,519)
90 FORMAT(1X,4E16,8)
98 FORMAT(15,A13,15,A8,F5.2,A3,F5.2,A13)
100 FORMAT(A2,(' ',12,' ')='E11.4,A2,E10.3,'*',A1,(' ',12,' '))
    &A2,E10.3,'*',A1,(' ',12,' ')='A2,E10.3,'*',A1,(' ',12,' ')
110 FORMAT(A2,E10.3,'*',A1,(' ',12,' ')='A2,E10.3,'*',A1,(' ',12,' '))
    &A2,E10.3,'*',A1,(' ',12,' ')='A2,E10.3,'*',A1,(' ',12,' ')
C=====
DATA (ISETID(K),K=1,15)/00,01,02,03,04,05,06,07,10,20,21,22,
&23,27,99/
DATA (IDSETI(K),K=1,100)/1,2,3,4,5,6,7,8,0,0,9*0,10,11,12,
&13,0,0,0,14,71*0,15/
ISETVAR=20
DATA (IW1(K,1),K=1,NSPAIR)/204,205,0,194,196,136,128,103,
&123,72,72,30,30,30,30,31,13,13,13,13,13,2,2,1,1,1,46,55,55,55,
&21,21,22,12,13,113,118,116,67,73,72,72,119,77,125,123,93,
&128,128,136,174,159,188,0,107,204,204,207,55,20,71,24,118,
&131,44,19,107,100,27,175,115,78,118,14,10,194,7,63,6,2,51,
&26,55,56,22,22,22,22,31,13,15,15/
DATA (IW1(K,2),K=1,NSPAIR)/209,209,1,201,201,137,137,126,
&126,73,73,31,31,31,31,13,13,13,13,13,2,2,1,1,56,56,56,56,
&22,22,22,15,15,126,126,126,73,73,73,126,126,126,126,
&137,137,137,137,201,201,201,201,209,209,209,209,56,22,73,
&137,137,137,56,22,201,209,209,209,126,126,137,126,73,209,126,
&137,137,209,56,56,56,22,22,22,22,31,13,15,15/
DATA (IW2(K,1),K=1,NSPAIR)/1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
&0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,1,3,1,0,0,0,0,4,0,0,
&3,0,1,0,2,0,2,0,2,13,2,0,6,0,0,0,2,1,5,3,1,1,0,0,5,
&3,1,1,53,10,2,1,1,0,2,0,0,0,0,0,0,0,0,0,0/
DATA (IW2(K,2),K=1,NSPAIR)/1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,
&4,2,1,1,1,1,4,4,22,15,1,15,1,1,1,1,54,22,15,31,13,2,
&4,31,13,2,3,31,13,2,4,31,13,2,4,31,13,2,4,73,73,15,
&56,22,15,201,201,15,56,22,15,137,73,73,201,201,73,209,201,
&209,201,31,13,2,4,31,13,2,4,15,15,2,4/
C=====
IMODE=MODE(1)
C=====
120 WRITE(06,140)
130 CALL DATIM(DATE1,TIME1)
140 FORMAT(72(1H=))
    CALL PTIME(TIMEIN)
    WRITE(06,*),
    & GALACTIC HYPERPLANES',
    CHMODE=CHTSH
    IF(IMODE.EQ.0) CHMODE=CHBAT
    WRITE(06,150)DATE1,TIME1,CHMODE
150 FORMAT(' RUN ON ',A8,' AT ',F6.2,' HOURS ON ',A7)
    WRITE(06,*),
C=====
WRITE(06,*), ' ENTER NUMBER & NAMES OF DATA FILES,WORDS/RECORD,
& V/N TO CLEARFILES',
DO 155 K=1,NFILMAX
155 FILIN(K)=FILECR
    & CHCF
    IF(IMODE.EQ.0)WRITE(06,*),NFIL,' ',(FILIN(1),I=1,MINO(NFILE,
&NFILMAX-1)),INMAX,' ',CHCF
    IF(NFILE.GT.NFILMAX-1) WRITE(06,*), ' CAN ACCEPT ONLY FIRST',

```

```

&NFILMAX=-1
& OF 'FILE' FILES OFFERED
REMCF='REMOVE CLEARFILES'
IF(CHCF.EQ.CHV.AND.IMODE.EQ.1)CALL CALLSS(REMCF)

C
I STAT=0
WRITE(06,*) 'NAME GALACTIC HYPERPLANE FILE'
FILEHO=FILECR
READ(05,*) FILEHO
IF(IMODE.EQ.0) WRITE(06,*) FILEHO
IF(FILEHO.EQ.FILECR) GO TO 257
OPEN(07,FILE=FILEHO,STATUS='OLD',Iostat=I STAT,ACCESS=
&'SEQUENTIAL',FORM='FORMATTED')
IF(I STAT.NE.0) THEN
WRITE(06,*) 'ERROR OPENING GALACTIC HYPERPLANE FILE.
& I STAT=',I STAT
STOP
END IF

C
WRITE(06,*) 'ENTER 0.1 TO NOTPERTURB,PERTURB INPUT DATA FILES'
READ(05,*) NPERT
IF(NPERT.NE.0) THEN
WRITE(06,*) 'ENTER RECORD OF FRACTIONS FOR +/- PERTURBATIONS'
READ(05,*) (PERTURB(I),I=1,INMAX)
IPERTT=0
DO 160 I=1,INMAX
IF(PERTURB(I).NE.0) IPERTT=IPERTT+1
160 CONTINUE
END IF

C
WRITE(06,*) 'NAME OUTPUT FILE FOR SET ASSIGNMENTS , , , ' TO
& PRINT
READ(05,*) FILEVOTE
IF(IMODE.EQ.0) WRITE(06,*) FILEVOTE
175 IOVOTE=06
I STAT=0
IF(FILEVOTE.EQ.FILECR) GO TO 210
INQUIRE(FILE=FILEVOTE,EXIST=EXIS)
IF(.NOT.EXIS) THEN
IOVOTE=08
OPEN(IOVOTE,FILE=FILEVOTE,STATUS='NEW',Iostat=I STAT,
& ACCESS='SEQUENTIAL',FORM='FORMATTED')
IF(I STAT.NE.0) THEN
WRITE(06,*) '
WRITE(06,*) 'ERROR OPENING SET ASSIGNMENT FILE.',
& I STAT=',I STAT
STOP
END IF
END IF
IF(EXIS) THEN
IF(IMODE.EQ.1) THEN
WRITE(06,*) '
WRITE(06,*) 'SET ASSIGNMENT FILE',FILEVOTE
& 'ALREADY EXISTS.'
WRITE(06,*) 'ENTER 1 TO OVERWRITE, 2 TO ENTER NEW NAME.'
180 READ(05,*) IOVERW
IF(IOVERW.NE.1.AND. IOVERW.NE.2) THEN
WRITE(06,*) 'MUST ENTER 1 OR 2. PLEASE TRY AGAIN.'
GO TO 180
END IF

```

```

      IF (IOVERW.EQ.1) THEN
        OPEN(IOVOTE,FILE=FILEVOTE,STATUS='OLD',IOSTAT=ISTAT)
        & ACCESS= 'SEQUENTIAL',FORM='FORMATTED')
        IF (ISTAT.NE.0) THEN
          WRITE(06,*) ' ERROR OPENING SET ASSIGNMENT FILE...'
          & ISTAT='ISTAT'
          STOP
        END IF
      END IF
      IF (IOVERW.EQ.2) THEN
        WRITE(06,*) ' ENTER NAME OF SET ASSIGNMENT FILE'
        & (' ' IF NONE)
        READ(05,*) FILEVOTE
        GO TO 175
      END IF
    END IF
    IF (IMODE.EQ.0) THEN
      CALL PTIME(TIMEOUT)
      WRITE(FILEVOTE,181) INT((TIMEOUT-TIMEIN)*3600.*1000.)
      181 FORMAT(18)
      WRITE(06,*) ' NAMING SET ASSIGNMENT FILE ' FILEVOTE
      OPEN(IOVOTE,FILE=FILEVOTE,STATUS='NEW',IOSTAT=ISTAT,
        & ACCESS= 'SEQUENTIAL',FORM='FORMATTED')
      IF (ISTAT.NE.0) THEN
        WRITE(06,*) ' ERROR OPENING SET ASSIGNMENT FILE'
        & ISTAT='ISTAT'
        STOP
      END IF
    END IF
  END IF
END IF
C
C READ FILEHO INTO TEMPM IN FORMAT: ISETID(I), ISETID(J),
C CONSTANTS FOR PAIR OF PLANES, INMAX COEFFICIENTS
210 DO 190 K=1, NSPAIR
  READ(07,*) (ITEMPM(K,J), J=1,2), (TEMPM(K,J), J=3, INMAX+4)
  190 CONTINUE
C=====
  DO 220 I=1, ISETMAX1
    NPINSET(I)=0
    ISETCOM(I)=0
    ISETCOM(I+ISETMAX1)=0
    ISETOUT(I)=0
    DO 215 J=1, ISETMAX
      DO 215 K=1, ICH
        IDCHOICE(J,K,I)=0
      215 CONTINUE
    220 CONTINUE
    IOUTPUT=0
    IOMAX=7
    NSETOUT=0
    NPOTOUT=0
    GO TO 290
  257 IF (IMODE.EQ.0) GO TO 259
    WRITE(06,260)
    GO TO 120
  259 STOP 10
  260 FORMAT(' IMPOSSIBLE INPUT INTERPRETED AS CUE TO RESTART RUN ')
  290 WRITE(06,*) ' ENTER N (LE', ISETMAX, ') THEN ID OF N SETS TO BE
    & EXCLUDED'
    READ(05,*) NSETOUT, (ISETOUT(I), I=1, MIN0(NSETOUT, ISETMAX1))

```

```

IF(IMODE.EQ.0)WRITE(06,*)NSETCOM, (ISETOUT(I), I=1, MIN0(NSETOUT
&, ISETMAX))
IF(NSETOUT.GT. ISETMAX)WRITE(06,295)ISETMAX
295 FORMAT( DATA IGNORED BEYOND DIMENSION LIMIT OF ', I4)
IF(NSETOUT.LT.0.AND. IMODE.EQ.1) GO TO 257
IF(NSETOUT.LT.0.AND. IMODE.EQ.0) STOP 11
C
WRITE(06,*)' ENTER N, THEN N PAIRS OF 1DS TO BE LABELED
& AS THE FIRST'
READ(05,*)NSETCOM, (ISETCOM(I), I=1, 2*MIN0(NSETCOM, ISETMAX))
IF(IMODE.EQ.0)
&WRITE(06,*)NSETCOM, (ISETCOM(I), I=1, 2*MIN0(NSETCOM, ISETMAX))
C
WRITE(06,*)' ENTER N THEN THE N HYPERPLANES TO BE USED
&(0 0 FOR ALL)'
NPLANIN=0
READ(05,*) NPLANIN, (IFLANE(I), I=1, MIN0(NPLANIN, NPLANMX))
C
WRITE(06,*)' ENTER 0 OR 1 TO NOTWRITE OR WRITE THE FOLLOWING
& OUTPUT OPTIONS'
WRITE(06,*)' HYP.PTS,V(T,ASS,ASP,ASC,HPT'
READ(05,*) IOUTPUT
IF(IDIGIT(IOUTPUT, IOMAX-6).NE.0) THEN
WRITE(06,*)' ENTER N (LE 20) THEN N RECORD NUMBERS FOR
& HYPERPLANE/POINT OUTPUT'
READ(05,*) NHPT, (IHP(I), I=1, MIN0(20, NHPT))
NHPT =MIN0(NHPT, 20)
END IF
C
C=====
IF(IDIGIT(IOUTPUT, IOMAX-7).NE.0) THEN
WRITE(06,*)' EQUATIONS FOR HYPERPLANES FROM GALACTIC'
DO 299 K=1, NSPAIR
WRITE(06,*)' TO SEPARATE SETS ', ITEMPM(K,1), ' & ',
& ITEMPM(K,2)
WRITE(06,*)' (TEMPM(K,J), J=3, INMXMX4)
299 CONTINUE
END IF
C=====
IER=0
ALPHA = .50
IF(IDIGIT(IOUTPUT, IOMAX-2).NE.0.OR.IDIGIT(IOUTPUT, IOMAX-3)
&.NE.0) THEN
WRITE(06,*)' P VOTE CONFIDENCE LEVEL =', ALPHA
END IF
DO 280 K=1, NSPAIR
CALL BELBIN(IW1(K,2), IW1(K,1), ALPHA, PHAT, PLOWER, PUPPER, IER)
W1(K,3)=PLOWER
IF(IER.NE.0) THEN
WRITE(06,*)' IER=', IER, ' FROM BELBIN. TRIALS.SUCCESSES = '
& .IW1(K,2), IW1(K,1), ' K=', K
W1(K,3)=0.
END IF
CALL BELBIN(IW2(K,2), IW2(K,1), ALPHA, PHAT, PLOWER, PUPPER, IER)
W2(K,3)=PLOWER
IF(IER.NE.0) THEN

```



```

      WRITE(IOVOTE,*) 'IER=',IER, ' FROM BELAIN. TRIALS,SUCCESSES='
      & .IW2(K,2),IW2(K,1), ' K=',K
      W2(K,3)=0.
    END IF
280 CONTINUE
    DO 291 I=1,ISETMAX
291 WSET(I)=0.
    DO 296 K=1,NSPAIR
      I1=IDSET1(ITEMPM(K,1)+1)
      I2=IDSET1(ITEMPM(K,2)+1)
      WSET(I1)=WSET(I1)+W1(K,3)
      WSET(I2)=WSET(I2)+W2(K,3)
296 CONTINUE
    IF(IDIGIT(IOUTPUT,IOMAX-2).NE.0.OR.IDIGIT(IOUTPUT,IOMAX-3)
      & .NE.0) THEN
      TEMP=ISETMAX-1
      WRITE(IOVOTE,*) ' '
      WRITE(IOVOTE,*) ' RELIABILITY INDEX PER SET '
      WRITE(IOVOTE,94) ' ID',(ISETID(K),K=1,ISETMAX)
      WRITE(IOVOTE,93) ' X',(WSET(K)/TEMP,K=1,ISETMAX)
    END IF
    =====
    NP=0
    NPTEMP=0
    NPT=0
    C NPTEMP=# PTS READ. NPT=# PTS NOT EXCLUDED BY USER, NP=# ANALYZED
    C NPFILE=PT# ON CURRENT FILE
    C NPAUS=PT# IN DATA BASE. IE. 1ST WORD OF RECORD
    DO305 K=1,NSPAIR
    DO305 J=1,8
      305 ISCOREP(K,J)=0
      DO 304 K=1,ISETMAX2
      DO 304 J=1,5
      DO 304 I=1,ISETMAX
      304 ISCOREST(I,K,J)=0
      DO 308 I=1,14
      308 ISCOREBP(I)=0
      IF(IDIGIT(IOUTPUT,IOMAX-2).EQ.0) GO TO 306
      WRITE(IOVOTE,*) ' '
      WRITE(IOVOTE,*) ' PT ACTID CANID VVOT NVOT
      & OVOT GVOT CVOT MX%O MX%G WVOTE BVOTE PVOTE '
      WRITE(IOVOTE,*) ' '
      306 NFILE=1
      CALL ATFILE(FILIN,NFILE)
      310 IF(IDIGIT(IOUTPUT,IOMAX-2).EQ.0.AND.
      & IDIGIT(IOUTPUT,IOMAX-3).EQ.0) GO TO 307
      WRITE(IOVOTE,*) ' '
      WRITE(IOVOTE,*) ' DATA FROM FILE ',FILIN(NFILE)
      307 NPFILE=0
      DO 343 K=1,ISETMAX1
      DO 343 J=1,5
      343 ISCORES(K,J)=0
      C
      320 READ(01,*,END=360)(BUFFER(L),L=1,INMAX)
      C
      NPTEMP=NPTEMP+1
      NPFILE=NPFILE+1
      NPABS= BUFFER(1)
      IPERT=0
      PERTS=1.

```

```

C      IF (IDIGIT(IOUTPUT, IOMAX-1), NE. 0) THEN
        WRITE(IOWTE,*)
        WRITE(IOWTE,*) POINT NUMBER, NPFILE
        WRITE(IOWTE,*)(BUFFER(L), L=1, INMAX)
      END IF

C      ITEMP=BUFFER(ISETVAR)
      IF (NSETOUT.EQ. 0.AND. NSETCOM.EQ. 0) GO TO 335
      IF (NSETOUT.LE. 0) GO TO 329
      DO 324 K=1, NSETOUT
        IF (ITEMP.EQ. ISETOUT(K)) GO TO 320
      324 CONTINUE
      329 IF (NSETCOM.LE. 0) GO TO 335
      DO 331 K=1, NSETCOM
        IF (ITEMP.EQ. ISETCOM(2*K)) ITEMP=ISETCOM(2*K-1)
      331 CONTINUE
C
      335 NPT=NPT+1
      NP=NP+1
      KO=IDSET1(ITEMP+1)
      NPINSET(KO)=NPINSET(KO)+1
      336 DO 340 K=1, ISETMAX
        DO 341 J=1, 5
          341 IVOTE(K, J)=0
        DO 342 J=1, 5
          342 VOTE(K, J)=0.
      340 CONTINUE
C
      K=0
      DO 357 I=1, ISETMAX-1
        DO 357 I2=I+1, ISETMAX
          K=K+1
          IF (NPLANIN, NE. 0) THEN
            DO 353 KK=1, NPLANIN
              IF (K.EQ. IPLANE(KK)) GO TO 354
            353 CONTINUE
            GO TO 357
          END IF
          354 TEMPV(K)=0.
          DO 355 J=1, INMAX
            IF (J.EQ. ISETVAR) GO TO 355
            TEMPER=0.
            IF (NPRT, NE. 0.AND. IPRT.EQ. J) TEMPER=PERTS*PERTURB(J)
            TEMPV(K)=TEMPV(K)+TEMPM(K, J+4)*BUFFER(J)*(1.+TEMPER)
          355 CONTINUE
            LHS1=TEMPM(K, 3)
            LHS2=TEMPM(K, 4)
            RHS=TEMPV(K)
            TEMPI=LHS1-RHS
            TEMPO=LHS1-LHS2
            K1=IDSET1(ITEMPM(K, 1)+1)
            K2=IDSET1(ITEMPM(K, 2)+1)
            IF (LHS1.LE. RHS.AND. LHS2.LT. RHS) THEN
              IVOTE(K1, 1)=IVOTE(K1, 1)+1
              IVOTE(K2, 2)=IVOTE(K2, 2)+1
              VOTE(K1, 3)=VOTE(K1, 3)+1.
              VOTE(K1, 4)=VOTE(K1, 4)+1.
              VOTE(K2, 4)=VOTE(K2, 4)-1.
            
```

```

VOTE(K1,5)=VOTE(K1,5)+W1(K,3)
VOTE(K2,5)=VOTE(K2,5)+W2(K,3)
IF(K0.EQ.K1) IWRITE(K1,6)=IWRITE(K1,6)+1
IF(K1.EQ.K0) ISCOREP(K,1)=ISCOREP(K,1)+1
IF(K2.EQ.K0) ISCOREP(K,5)=ISCOREP(K,5)+1
END IF
IF(LHS1.GE.RHS.AND.LHS2.GE.RHS) THEN
  IWRITE(K1,2)=IWRITE(K1,2)+1
  IWRITE(K2,1)=IWRITE(K2,1)+1
  VOTE(K2,3)=VOTE(K2,3)+1
  VOTE(K1,4)=VOTE(K1,4)+1
  VOTE(K2,4)=VOTE(K2,4)+1
  VOTE(K1,5)=VOTE(K1,5)+W1(K,3)
  VOTE(K2,5)=VOTE(K2,5)+W2(K,3)
  IF(K0.EQ.K2) IWRITE(K2,6)=IWRITE(K2,6)+1
  IF(K1.EQ.K0) ISCOREP(K,2)=ISCOREP(K,2)+1
  IF(K2.EQ.K0) ISCOREP(K,6)=ISCOREP(K,6)+1
END IF
IF(LHS1.LE.RHS.AND.LHS2.LE.RHS) THEN
  IWRITE(K1,3)=IWRITE(K1,3)+1
  IWRITE(K2,3)=IWRITE(K2,3)+1
  IF(TEMPO.NE.0.) THEN
    VOTE(K1,3)=VOTE(K1,3)+AMIN1(1.,TEMP1/TEMPO)
    VOTE(K2,3)=VOTE(K2,3)+AMIN1(1.,TEMP2/TEMPO)
    VOTE(K1,1)=AMAX1(VOTE(K1,1),TEMP1/TEMPO)
    VOTE(K2,1)=AMAX1(VOTE(K2,1),TEMP2/TEMPO)
    VOTE(K1,4)=VOTE(K1,4)+(TEMP1-TEMP2)/TEMPO
    VOTE(K2,4)=VOTE(K2,4)+(TEMP2-TEMP1)/TEMPO
    VOTE(K1,5)=VOTE(K1,5)+(TEMP1-TEMP2)/TEMPO*W1(K,3)
    VOTE(K2,5)=VOTE(K2,5)+(TEMP2-TEMP1)/TEMPO*W2(K,3)
  END IF
  IF(K1.EQ.K0) ISCOREP(K,3)=ISCOREP(K,3)+1
  IF(K2.EQ.K0) ISCOREP(K,7)=ISCOREP(K,7)+1
END IF
IF(LHS1.GT.RHS.AND.LHS2.LT.RHS) THEN
  IWRITE(K1,4)=IWRITE(K1,4)+1
  IWRITE(K2,4)=IWRITE(K2,4)+1
  IF(TEMPO.NE.0.) THEN
    VOTE(K1,3)=VOTE(K1,3)+AMIN1(1.,TEMP2/TEMPO)
    VOTE(K2,3)=VOTE(K2,3)+AMIN1(1.,TEMP1/TEMPO)
    VOTE(K1,2)=AMAX1(VOTE(K1,2),TEMP1/TEMPO)
    VOTE(K2,2)=AMAX1(VOTE(K2,2),TEMP2/TEMPO)
    VOTE(K1,4)=VOTE(K1,4)+(TEMP2-TEMP1)/TEMPO
    VOTE(K2,4)=VOTE(K2,4)+(TEMP1-TEMP2)/TEMPO
    VOTE(K1,5)=VOTE(K1,5)+(TEMP2-TEMP1)/TEMPO*W1(K,3)
    VOTE(K2,5)=VOTE(K2,5)+(TEMP1-TEMP2)/TEMPO*W2(K,3)
  END IF
  IF(K1.EQ.K0) ISCOREP(K,4)=ISCOREP(K,4)+1
  IF(K2.EQ.K0) ISCOREP(K,8)=ISCOREP(K,8)+1
END IF
IF(10*IGIT(10*OUTPUT,IOMAX-6).EQ.0) GO TO 357
DO 301 I=1,NHPT
  IF(NPTEMP.EQ.IHPT(I)) GO TO 302
301 CONTINUE
  TEMPO=0.
  GO TO 357
302 ITEMPO=1.
  IF(K.EQ.1) WRITE(10*VOTE,*)
  IF(K.EQ.1)WRITE(10*VOTE,*) HYPERPLANE VOTES FOR POINT #

```

C

```

&NPABS, RECORD, NPTEMP
IF(K.EQ.1)WRITE(IOVOTE,*)' HYP# SETID
& NORMAL TO PLANES
CHB1=CHLB
CHB2=CHRB
IF(LHS1.GT.LHS2) THEN
  CHB1=CHRB
  CHB2=CHLB
END IF
TEMP1=AMIN1(LHS1,LHS2)
TEMP2=AMAX1(LHS1,LHS2)
IF(TEMP1.LE.RHS.AND.TEMP2.LE.RHS) WRITE(IOVOTE,71)
&K,ITEMPM(K,2),CHB1,TEMP1,TEMP2,CHB2,RHS,ITEMPM(K,1)
IF(TEMP1.LT.RHS.AND.TEMP2.GT.RHS) WRITE(IOVOTE,72)
&K,ITEMPM(K,2),CHB1,TEMP1,RHS,TEMP2,CHB2,ITEMPM(K,1)
IF(TEMP1.EQ.RHS.AND.TEMP2.EQ.RHS) GO TO 357
IF(TEMP1.GE.RHS.AND.TEMP2.GE.RHS) WRITE(IOVOTE,73)
&K,ITEMPM(K,2),RHS,CHB1,TEMP1,TEMP2,CHB2,ITEMPM(K,1)

C 357 CONTINUE
IF(TEMPO.NE.0.) WRITE(IOVOTE,*)'
C
TEMPO=FLOAT(ISETMAX-1)
DO 351 K=1,ISETMAX
  IOVOTE(K,5)=IOVOTE(K,1)-IOVOTE(K,2)+IOVOTE(K,3)+IOVOTE(K,4)
351 CONTINUE
C=====
KMAX1=1
KMAX2=1
KMAX3=1
KMAX4=1
KKMAX1=0
KKMAX2=0
KKMAX3=0
KKMAX4=0
KKMAX5=0
KOUNT1=0
KOUNT2=0
KOUNT3=0
KOUNT4=0
KOUNT5=0
K2MAX1=0
K2MAX2=0
K2MAX3=0
K2MAX4=0
K2MAX5=0
DO 356 K=1,ISETMAX
  IF(IOVOTE(K,1).GT.IOVOTE(KMAX1,1)) KMAX1=K
  IF(IOVOTE(K,1).EQ.IOVOTE(KMAX1,1).AND.K.NE.KMAX1)KKMAX1=1
  IF(K.EQ.KMAX1) KKMAX1=0
  IF(IOVOTE(K,1).GT.IOVOTE(KO,1)) KOUNT1=KOUNT1+1
  IF(IOVOTE(K,1).LT.IOVOTE(KMAX1,1)) THEN
    IF(K2MAX1.EQ.0) K2MAX1=K
    IF(IOVOTE(K,1).GT.IOVOTE(K2MAX1,1)) K2MAX1=K
  END IF
  IF(IOVOTE(K,5).GT.IOVOTE(KMAX2,5)) KMAX2=K
  IF(IOVOTE(K,5).EQ.IOVOTE(KMAX2,5).AND.K.NE.KMAX2)KKMAX2=1
  IF(K.EQ.KMAX2)KKMAX2=0
  IF(IOVOTE(K,5).GT.IOVOTE(KO,5)) KOUNT2=KOUNT2+1
  IF(IOVOTE(K,5).LT.IOVOTE(KMAX2,5)) THEN

```

```

IF (K2MAX2.EQ.0) K2MAX2=K
IF (IVOTE(K,5).GT.IVOTE(K2MAX2,5)) K2MAX2=K
END IF
IF (VOTE(K,3).GT.VOTE(KMAX3,3)) KMAX3=K
IF (VOTE(K,3).EQ.VOTE(KMAX3,3).AND.K.NE.KMAX3) KMAX3=1
IF (K.EQ.KMAX3) KMAX3=0
IF (VOTE(K,3).GT.VOTE(K0,3)) KOUNT3=KOUNT3+1
IF (IVOTE(K,3).LT.IVOTE(KMAX3,3)) THEN
IF (K2MAX3.EQ.0) K2MAX3=K
IF (IVOTE(K,3).GT.IVOTE(K2MAX3,3)) K2MAX3=K
END IF
IF (VOTE(K,4).GT.VOTE(KMAX4,4)) KMAX4=K
IF (VOTE(K,4).EQ.VOTE(KMAX4,4).AND.K.NE.KMAX4) KMAX4=1
IF (K.EQ.KMAX4) KMAX4=0
IF (VOTE(K,4).GT.VOTE(K0,4)) KOUNT4=KOUNT4+1
IF (VOTE(K,4).LT.VOTE(KMAX4,4)) THEN
IF (K2MAX4.EQ.0) K2MAX4=K
IF (IVOTE(K,4).GT.VOTE(K2MAX4,4)) K2MAX4=K
END IF
IF (VOTE(K,5).GT.VOTE(KMAX5,5)) KMAX5=K
IF (VOTE(K,5).EQ.VOTE(KMAX5,5).AND.K.NE.KMAX5) KMAX5=1
IF (K.EQ.KMAX5) KMAX5=0
IF (VOTE(K,5).GT.VOTE(K0,5)) KOUNT5=KOUNT5+1
IF (VOTE(K,5).LT.VOTE(KMAX5,5)) THEN
IF (K2MAX5.EQ.0) K2MAX5=K
IF (IVOTE(K,5).GT.VOTE(K2MAX5,5)) K2MAX5=K
END IF
356 CONTINUE
IF (IVOTE(K0,1).EQ.IVOTE(KMAX1,1)) THEN
IF (KMAX1.EQ.0) ISCORES(1,1)=ISCORES(1,1)+1
IF (KMAX1.EQ.1) ISCORES(2,1)=ISCORES(2,1)+1
IF (KMAX1.EQ.0) ISCOREST(K0,1,1)=ISCOREST(K0,1,1)+1
IF (KMAX1.EQ.1) ISCOREST(K0,2,1)=ISCOREST(K0,2,1)+1
END IF
IF (IVOTE(K0,5).EQ.IVOTE(KMAX2,5)) THEN
IF (KMAX2.EQ.0) ISCORES(1,2)=ISCORES(1,2)+1
IF (KMAX2.EQ.1) ISCORES(2,2)=ISCORES(2,2)+1
IF (KMAX2.EQ.0) ISCOREST(K0,1,2)=ISCOREST(K0,1,2)+1
IF (KMAX2.EQ.1) ISCOREST(K0,2,2)=ISCOREST(K0,2,2)+1
END IF
IF (VOTE(K0,3).EQ.VOTE(KMAX3,3)) THEN
IF (KMAX3.EQ.0) ISCORES(1,3)=ISCORES(1,3)+1
IF (KMAX3.EQ.1) ISCORES(2,3)=ISCORES(2,3)+1
IF (KMAX3.EQ.0) ISCOREST(K0,1,3)=ISCOREST(K0,1,3)+1
IF (KMAX3.EQ.1) ISCOREST(K0,2,3)=ISCOREST(K0,2,3)+1
END IF
IF (VOTE(K0,4).EQ.VOTE(KMAX4,4)) THEN
IF (KMAX4.EQ.0) ISCORES(1,4)=ISCORES(1,4)+1
IF (KMAX4.EQ.1) ISCORES(2,4)=ISCORES(2,4)+1
IF (KMAX4.EQ.0) ISCOREST(K0,1,4)=ISCOREST(K0,1,4)+1
IF (KMAX4.EQ.1) ISCOREST(K0,2,4)=ISCOREST(K0,2,4)+1
END IF
IF (VOTE(K0,5).EQ.VOTE(KMAX5,5)) THEN
IF (KMAX5.EQ.0) ISCORES(1,5)=ISCORES(1,5)+1
IF (KMAX5.EQ.1) ISCORES(2,5)=ISCORES(2,5)+1
IF (KMAX5.EQ.0) ISCOREST(K0,1,5)=ISCOREST(K0,1,5)+1
IF (KMAX5.EQ.1) ISCOREST(K0,2,5)=ISCOREST(K0,2,5)+1
END IF
IF (KOUNT1.GT.0) THEN
ISCORES(KOUNT1+2,1)=ISCORES(KOUNT1+2,1)+1

```

```

ISCOREST(K0,KOUNT1+2,1)=ISCOREST(K0,KOUNT1+2,1)+1
END IF
IF (KOUNT2.GT.0) THEN
  ISCORES(KOUNT2+2,2)=ISCORES(KOUNT2+2,2)+1
  ISCOREST(K0,KOUNT2+2,2)=ISCOREST(K0,KOUNT2+2,2)+1
END IF
IF (KOUNT3.GT.0) THEN
  ISCORES(KOUNT3+2,3)=ISCORES(KOUNT3+2,3)+1
  ISCOREST(K0,KOUNT3+2,3)=ISCOREST(K0,KOUNT3+2,3)+1
END IF
IF (KOUNT4.GT.0) THEN
  ISCORES(KOUNT4+2,4)=ISCORES(KOUNT4+2,4)+1
  ISCOREST(K0,KOUNT4+2,4)=ISCOREST(K0,KOUNT4+2,4)+1
END IF
IF (KOUNT5.GT.0) THEN
  ISCORES(KOUNT5+2,5)=ISCORES(KOUNT5+2,5)+1
  ISCOREST(K0,KOUNT5+2,5)=ISCOREST(K0,KOUNT5+2,5)+1
END IF

ITEMP1=0
IF (VOTE(KMAX4,4).EQ.VOTE(KMAX5,4).AND.VOTE(KMAX4,5).EQ.
&VOTE(KMAX5,5)) ITEMP1=1
ITEMP2=0
IF (VOTE(K0,4).EQ.VOTE(KMAX4,4)) ITEMP2=1
ITEMP3=0
IF (VOTE(K0,5).EQ.VOTE(KMAX5,5)) ITEMP3=1
IF (ITEMP1.EQ.1) THEN
  IF (ITEMP2.EQ.1.AND.ITEMP3.EQ.1) ISCOREBP(1)=ISCOREBP(1)+1
  IF (ITEMP2.EQ.0.OR.ITEMP3.EQ.0) ISCOREBP(2)=ISCOREBP(2)+1
END IF
IF (ITEMP1.EQ.0) THEN
  IF (ITEMP2.EQ.1.AND.ITEMP3.EQ.0) ISCOREBP(3)=ISCOREBP(3)+1
  IF (ITEMP2.EQ.0.AND.ITEMP3.EQ.1) ISCOREBP(4)=ISCOREBP(4)+1
  IF (ITEMP2.EQ.0.AND.ITEMP3.EQ.0) ISCOREBP(5)=ISCOREBP(5)+1
END IF
ITEMP1=6+3*MIN0(KOUNT4,2)+MIN0(KOUNT5,2)
ISCOREBP(ITEMP1)=ISCOREBP(ITEMP1)+1

IF (IDIGIT(IOUTPUT,IOMAX-2).NE.0) THEN
  IF (IPERT.GT.0.AND.PERTS.EQ.-1.) WRITE(IG.OTE,*)
& ' + PERTURBATION OF VARIABLE ,IPERT
  IF (IPERT.GT.0.AND.PERTS.EQ.+1.) WRITE(IG.OTE,*)
& ' - PERTURBATION OF VARIABLE ,IPERT
END IF

ITEMP0=1
ITEMP1=IVOTE(ITEMP0,1)
ITEMP2=0
ITEMP3=0
DO 358 K=ITEMP0+1,ISETMAX
  IF (IVOTE(K,1).GT.ITEMP1) THEN
    ITEMP0=K
    ITEMP1=IVOTE(ITEMP0,1)
    ITEMP2=0
    GO TO 358
  END IF
  IF (IVOTE(K,1).EQ.ITEMP1) ITEMP2=1
358 CONTINUE
  IK=ITEMP0
  ASSIGN 371 TO IRETURN

```

```

GO TO 400
371 IF (IDIGIT(IOUTPUT,IOMAX-2).NE.0)
&WRITE(IOVOTE,80)NGBS,ITEMP,ISETID(IITEMP),
&IWRITE(IITEMP,1),CH1,(IVOTE(IITEMP,J),J=2,5),CH2,
&(VOTE(IITEMP,J),J=1,3),CH3,VOTE(IITEMP,4),CH4
&,VOTE(IITEMP,5),CH5
IF (IITEMP.EQ.K0) ITEMP3=1
IF (IITEMP2.EQ.0) GO TO 370
DO 359 K=IITEMP+1,ISETMAX
IF (IVOTE(K,1).EQ.IITEMP1) THEN
IK=K
ASSIGN 372 TO IRETURN
GO TO 400
372 IF (IDIGIT(IOUTPUT,IOMAX-2).NE.0)
& WRITE(IOVOTE,81)CHBL,ITEMP,ISETID(K),
& ,VOTE(K,1),CH1,(IVOTE(K,J),J=2,5),CH2,(VOTE(K,J),J=1,3)
& ,CH3,VOTE(K,4),CH4,VOTE(K,5),CH5
IF (K.EQ.K0) ITEMP3=1
END IF
359 CONTINUE
370 IF (ITEMP3.NE.0) GO TO 375
IF (K0.GT.ISETMAX) GO TO 375
IK=K0
ASSIGN 373 TO IRETURN
GO TO 400
373 IF (IDIGIT(IOUTPUT,IOMAX-2).NE.0)
&WRITE(IOVOTE,81)CHBL,ITEMP,ITEMP,IVOTE(K0,1),CH1,
&(IVOTE(K0,J),J=2,5),CH2,(VOTE(K0,J),J=1,3),CH3,
&VOTE(K0,4),CH4,VOTE(K0,5),CH5
C
375 IF (IDIGIT(IOUTPUT,IOMAX-2).EQ.0) GO TO 381
DO 380 K=1,ISETMAX
IF (IVOTE(K,1).EQ.ITEMP1.OR.K.EQ.K0) GO TO 380
IK=K
ASSIGN 374 TO IRETURN
GO TO 400
374 IF (NPLANIN.NE.0) THEN
DO 377 J=1,5
IF (IVOTE(K,J).NE.0) GO TO 379
377 CONTINUE
DO 378 J=1,4
IF (VOTE(K,J).NE.0.) GO TO 379
378 CONTINUE
GO TO 380
END IF
379 WRITE(IOVOTE,82)CHBL,CHBL,ISETID(K),
&IVOTE(K,1),CH1,(IVOTE(K,J),J=2,5),CH2,(VOTE(K,J),J=1,3)
&,CH3,VOTE(K,4),CH4,VOTE(K,5),CH5
380 CONTINUE
C
381 IF (IDIGIT(IOUTPUT,IOMAX-5).EQ.0) G) TO 389
K3=0
K4=0
K5=0
K6=0
K7=0
K8=0
DO 383 K=1,ISETMAX
IB=2

```

```

IF(VOTE(K,4).EQ.VOTE(KMAX4,4)) IB=0
IF(VOTE(K,4).EQ.VOTE(K2MAX4,4) AND K2MAX4.NE.0) IB=1
IP=2
IF(VOTE(K,5).EQ.VOTE(KMAX5,5)) IP=0
IF(VOTE(K,5).EQ.VOTE(K2MAX5,5) AND K2MAX5.NE.0) IP=1
IF(IB.EQ.0.AND.IP.EQ.0) THEN
  IDCHOICE(K,1,K0)=IDCHOICE(K,1,K0)+1
  IDCHOICE(K,6,K0)=IDCHOICE(K,6,K0)+1
  IDCHOICE(K,7,K0)=IDCHOICE(K,7,K0)+1
  IDCHOICE(K,8,K0)=IDCHOICE(K,8,K0)+1
  IDCHOICE(K,9,K0)=IDCHOICE(K,9,K0)+1
  IDCHOICE(K,10,K0)=IDCHOICE(K,10,K0)+1
  IDCHOICE(K,11,K0)=IDCHOICE(K,11,K0)+1
END IF
IF(IB.EQ.0) IDCHOICE(K,2,K0)=IDCHOICE(K,2,K0)+1
IF(IP.EQ.0) IDCHOICE(K,3,K0)=IDCHOICE(K,3,K0)+1
IF(K.EQ.K0) THEN
  IDCHOICE(K,4,1SETMAX1)=IDCHOICE(K,4,1SETMAX1)+1
  IDCHOICE(K,5,1SETMAX1)=IDCHOICE(K,5,1SETMAX1)+1
  IF(IB.NE.2.AND.IP.NE.2) IDCHOICE(K,4,K0)=IDCHOICE(K,4,K0)+1
  IF(IB.NE.2.OR.IP.NE.2) IDCHOICE(K,5,K0)=IDCHOICE(K,5,K0)+1
END IF
IF(IB.EQ.0.AND.IP.EQ.0) K3=1
IF(IB.EQ.1.AND.IP.EQ.1) K4=K
IF(IB.EQ.0.AND.IP.EQ.1) K5=K
IF(IB.EQ.1.AND.IP.EQ.0) K6=K
IF(IP.EQ.0.AND.K7.EQ.0) K7=K
IF(IB.EQ.0.AND.K8.EQ.0) K8=K
383 CONTINUE
IF(K3.EQ.0) THEN
  IF(K4.GT.0) IDCHOICE(K4,6,K0)=IDCHOICE(K4,6,K0)+1
  IF(K4.NE.0) IDCHOICE(K4,7,K0)=IDCHOICE(K4,7,K0)+1
  IF(K4.EQ.0) THEN
    IF(K5.NE.0) IDCHOICE(K5,7,K0)=IDCHOICE(K5,7,K0)+1
    IF(K5.EQ.0.AND.K6.NE.0) IDCHOICE(K6,7,K0)=IDCHOICE(K6,7,K0)+1
  END IF
  IF(K5.NE.0) IDCHOICE(K5,8,K0)=IDCHOICE(K5,8,K0)+1
  IF(K5.EQ.0.AND.K6.NE.0) IDCHOICE(K6,8,K0)=IDCHOICE(K6,8,K0)+1
  IF(K5.EQ.0.AND.K6.EQ.0.AND.K4.NE.0) IDCHOICE(K4,8,K0)=
    IDCHOICE(K4,8,K0)+1
  IF(K5.EQ.0.AND.K6.EQ.0.AND.K4.EQ.0.AND.K7.NE.0)
    IDCHOICE(K7,8,K0)=IDCHOICE(K7,8,K0)+1
  IF(K6.NE.0) IDCHOICE(K6,9,K0)=IDCHOICE(K6,9,K0)+1
  IF(K6.EQ.0.AND.K5.NE.0) IDCHOICE(K5,9,K0)=IDCHOICE(K5,9,K0)+1
  IF(K6.EQ.0.AND.K5.EQ.0.AND.K4.NE.0) IDCHOICE(K4,9,K0)=
    IDCHOICE(K4,9,K0)+1
  IF(K6.EQ.0.AND.K5.EQ.0.AND.K4.EQ.0.AND.K7.NE.0)
    IDCHOICE(K7,9,K0)=IDCHOICE(K7,9,K0)+1
  IF(K5.NE.0) IDCHOICE(K5,10,K0)=IDCHOICE(K5,10,K0)+1
  IF(K5.NE.0) IDCHOICE(K5,11,K0)=IDCHOICE(K5,11,K0)+1
  IF(K5.EQ.0.AND.K8.NE.0) IDCHOICE(K8,11,K0)=IDCHOICE(K8,11,K0)+1
END IF
389 IF(NPERT.EQ.0) GO TO 320
PERTS=PERTS
IF(PERTS.GT.0.) GO TO 336
IF(IPERT.EQ.INMAX) GO TO 320
DO 401 I=IPERT+1,INMAX
  IF(PERTURB(I).EQ.0) GO TO 401
  IPERT=I

```


GO TO 336
401 CONTINUE
GO TO 320

C
C
C

400 CH1=CHBL1
CH2=CHBL1
CH3=CHBL1
CH4=CHBL1
CH5=CHBL1

IF(I(VOTE(IK,1)).GE.I(VOTE(KMAX1,1)))CH1=CHASK
IF(I(VOTE(IK,5)).GE.I(VOTE(KMAX2,5)))CH2=CHASK
IF(VOTE(IK,3).GE.VOTE(KMAX3,3))CH3=CHASK
IF(VOTE(IK,4).GE.VOTE(KMAX4,4))CH4=CHASK
IF(VOTE(IK,5).GE.VOTE(KMAX5,5))CH5=CHASK
GO TO IRETURN,(371,372,373,374)

C
C
C

360 IF(IDIGIT(IOUTPUT,IOMAX-3).EQ.0) GO TO 363
WRITE(I(VOTE,*))
WRITE(I(VOTE,*)) PLACEMENT OF IDENTIFIED SETS'
WRITE(I(VOTE,*)) YVOTE CVOTE WVOTE
8 BVOTE PVOTE
WRITE(I(VOTE,88)) UNIQUE 1ST PLACE', (ISCORES(1,J),J=1,5)
WRITE(I(VOTE,88)) TIED 1ST PLACE', (ISCORES(2,J),J=1,5)
DO 361 K=3,ISETMAX1
WRITE(I(VOTE,89)K-2,' WITH > VOTES', (ISCORES(K,J),J=1,5)
361 CONTINUE

C

363 NFILE=NFILE+1
CALL ATFILE(FILIN,NFILE)
IF(NFILE.NE.0) GO TO 310

C

CLOSE(07,STATUS='KEEP')
IF(I(VOTE.EQ.08) CLOSE(08,STATUS='KEEP')

C

IF(IDIGIT(IOUTPUT,IOMAX-3).EQ.0) GO TO 410

WRITE(I(VOTE,*))
WRITE(I(VOTE,*)) DISCRIMINATION SUCCESS BY SET, USING B VOTES'
WRITE(I(VOTE,*)) SETS
WRITE(I(VOTE,91)) VOTES', (ISETID(J),J=1,ISETMAX)
WRITE(I(VOTE,92)0,(ISCORES(K,1,4)+ISCORES(K,2,4),K=1,ISETMAX)
DO 382 I=3,ISETMAX1
WRITE(I(VOTE,92)I-2,(ISCORES(K,I,4),K=1,ISETMAX)

C

382 CONTINUE
DO 391 K=1,ISETMAX
DO 391 I=1,ISETMAX1
391 ISCORES(K,ISETMAX2,4)=ISCORES(K,I,4)+ISCORES(K,ISETMAX2,4)
WRITE(I(VOTE,91)) TOTAL', (ISCORES(K,ISETMAX2,4),K=1,ISETMAX)

C

WRITE(I(VOTE,*))
WRITE(I(VOTE,*)) DISCRIMINATION SUCCESS BY SET, USING P VOTES'
WRITE(I(VOTE,*)) SETS
WRITE(I(VOTE,91)) VOTES', (ISETID(J),J=1,ISETMAX)
WRITE(I(VOTE,92)0,(ISCORES(K,1,5)+ISCORES(K,2,5),K=1,ISETMAX)
DO 387 I=3,ISETMAX1
WRITE(I(VOTE,92)I-2,(ISCORES(K,I,5),K=1,ISETMAX)

```

387 CONTINUE
DO 397 K=1,ISETMAX
DO 397 I=1,ISETMAX
397 ISCOREST(K,ISETMAX+1,5)=ISCOREST(K,I,5)+ISCOREST(K,ISETMAX2,5)
WRITE(I0VOTE,91) ISCOREST(K,ISETMAX2,4),K=1,ISETMAX
C
WRITE(I0VOTE,*)
DISCRIMINATION SUCCESS USING B & P VOTES JOINTLY
WRITE(I0VOTE,*) P & B AGREE, ON IDENTIFIED SET, ISCOREBP(1)
WRITE(I0VOTE,*) B NOT P SELECTS IDENTIFIED SET, ISCOREBP(3)
WRITE(I0VOTE,*) P NOT B SELECTS IDENTIFIED SET, ISCOREBP(4)
WRITE(I0VOTE,*) P & B DISAGREE, ON OTHER SETS, ISCOREBP(5)
WRITE(I0VOTE,*) B & P AGREE, ON ONE OTHER SET, ISCOREBP(2)
WRITE(I0VOTE,*) TOTAL, ISCOREBP(1)+
&ISCOREBP(2)+ISCOREBP(3)+ISCOREBP(4)+ISCOREBP(5)
WRITE(I0VOTE,*) B & P VOTE RANK OF IDENTIFIED SET,
&PRANK, BRANK I= 2= OTHER=
DO 398 I=1,2
WRITE(I0VOTE,*) I, ISCOREBP(5+3*(I-1)+J),J=1,3)
398 CONTINUE
WRITE(I0VOTE,*) OTHER, (ISCOREBP(5+3*2+J),J=1,3)
C
IF(IDIGIT(IOUTPUT,IGMAX-5).EQ.0) GO TO 414
410 WRITE(I0VOTE,*)
WRITE(I0VOTE,*) SUCCESS OF SEVERAL CHOICE RULES
DO 399 K=1,ICH
ITEMP1=0
ITEMP2=0
DO 394 I=1,ISETMAX
IF(K.EQ.4.OR.K.EQ.5) GO TO 393
DO 395 J=1,ISETMAX
IDCHOICE(I,K,ISETMAX1)=IDCHOICE(I,K,ISETMAX1)+IDCHOICE(I,K,J)
395 CONTINUE
ITEMP1=ITEMP1+IDCHOICE(I,K,I)
ITEMP2=ITEMP2+IDCHOICE(I,K,ISETMAX1)
PLOWER=0
IF(IDCHOICE(I,K,ISETMAX1).EQ.0) GO TO 392
CALL BELBIN(IDCHOICE(I,K,ISETMAX1),IDCHOICE(I,K,I),ALPHA
&,PHAT,PLOWER,PUPPER,IER)
IF(IER.NE.0) THEN
WRITE(I0VOTE,*) IER, IER, FROM BELBIN. TRIALS,SUCCESSSES=
& IDCHOICE(I,K,ISETMAX1),IDCHOICE(I,K,I)
PLOWER=0.
END IF
392 XCHOICE(I,K,ISETMAX1)=PLOWER
394 CONTINUE
WRITE(I0VOTE,*)
IF(K.EQ.1)WRITE(I0VOTE,*) CHOOSE SET WITH B1=P1, ELSE NO CHOICE
IF(K.EQ.2)WRITE(I0VOTE,*) CHOOSE SET WITH B1
IF(K.EQ.3)WRITE(I0VOTE,*) CHOOSE SET WITH P1
IF(K.EQ.4)WRITE(I0VOTE,*) CHOOSE SETS WITH (B1 OR B2)&(P1 OR P2)
IF(K.EQ.5)WRITE(I0VOTE,*) CHOOSE SETS WITH B1 OR B2 OR P1 OR P2
IF(K.EQ.6)WRITE(I0VOTE,*) CHOOSE SET WITH B1=P1, ELSE WITH B2=P2
&, ELSE NO CHOICE
IF(K.EQ.7)WRITE(I0VOTE,*) CHOOSE SET WITH B1=P1, ELSE B2=P2,
& ELSE B2=P1, ELSE B1=P2, ELSE NO CHOICE
IF(K.EQ.8)WRITE(I0VOTE,*) CHOOSE SET WITH B1=P1, ELSE B2=P1,
& ELSE B1=P2, ELSE B2=P2, ELSE P1

```

```

IF(K.EQ.9)WRITE(IOVOTE,*)' CHOOSE SET WITH B1=P1, ELSE B1=P2,
& ELSE B2=P1, ELSE B2=P2, ELSE P1'
IF(K.EQ.10)WRITE(IOVOTE,*)' CHOOSE SET WITH B1=P1, ELSE B1=P2,
& ELSE NO CHOICE'
IF(K.EQ.11)WRITE(IOVOTE,*)' CHOOSE SET WITH B1=P1, ELSE B1=P2,
& ELSE B1'
WRITE(IOVOTE,*)'
&
WRITE(IOVOTE,96)' CHOICE', (ISETID(J),J=1,ISETMAX),' SUCCESS'
DO 396 I=1,ISETMAX
WRITE(IOVOTE,95)ISETID(I),(IDCHOICE(I,K,J),J=1,ISETMAX)
&XCHOICE(I,K,ISETMAX1)
396 CONTINUE
CALL BELBIN(ITEMP2,TEMP1,ALPHA,PHAT,PLOWER,PUPPER,IER)
WRITE(IOVOTE,98) ITEMP1,' SUCCESSES IN',ITEMP2,' TRIALS,'
&PLOWER,' TO',PUPPER,' PROB SUCCESS'
399 CONTINUE
C=====
414 IF((IDIGIT(OUTPUT,IOMAX-4).EQ.0) GO TO 9000
WRITE(IOVOTE,*)'
WRITE(IOVOTE,*)' DISCRIMINATION SUCCESS BY HYPERPLANE'
WRITE(IOVOTE,*)' HYPERPLANE POINTS IN THE FIRST SET
& POINTS IN THE SECOND SET'
WRITE(IOVOTE,*)' #SET1 SET2 NVOTE OVOTE GVOTE V%T
& NVOTE NVOTE OVOTE GVOTE V%T'
K=0
DO 385 I1=1,ISETMAX-1
DO 385 I2=I1+1,ISETMAX
K=K+1
K1=IDSET1(ITEMPM(K,1)+1)
K2=IDSET1(ITEMPM(K,2)+1)
TEMP1=-1.
ITEMP1=ISCOREP(K,1)+ISCOREP(K,2)+ISCOREP(K,3)+ISCOREP(K,4)
IF(ITEMP1.NE.0)TEMP1=FLOAT(ISCOREP(K,1))/FLOAT(ITEMP1)*100.
TEMP2=-1.
ITEMP2=ISCOREP(K,5)+ISCOREP(K,6)+ISCOREP(K,7)+ISCOREP(K,8)
IF(ITEMP2.NE.0)TEMP2=FLOAT(ISCOREP(K,5))/FLOAT(ITEMP2)*100.
IF(ITEMP1.NE.0.AND.ITEMP2.NE.0)
&WRITE(IOVOTE,85) K,ITEMPM(K,1),ITEMPM(K,2),
&(ISCOREP(K,J),J=1,4),TEMP1,(ISCOREP(K,J),J=5,8),TEMP2
IF(ITEMP1.NE.0.AND.ITEMP2.EQ.0) WRITE(IOVOTE,86) K,ITEMPM(K,1)
&ITEMPM(K,2),(ISCOREP(K,J),J=1,4),TEMP1
IF(ITEMP1.EQ.0.AND.ITEMP2.NE.0) WRITE(IOVOTE,87) K,ITEMPM(K,1)
&ITEMPM(K,2),(ISCOREP(K,J),J=5,8),TEMP2
385 CONTINUE
C
WRITE(IOVOTE,*)'
WRITE(IOVOTE,*)' PLANE# SET#1 RELINDEX SET
&#2 RELINDEX'
DO 415 K=1,NSPAIR
WRITE(IOVOTE,*)K,ITEMPM(K,1),W1(K,3),ITEMPM(K,2),W2(K,3)
415 CONTINUE
C=====
C TERMINATING RUN PROCEDURE
9000 WRITE(IOVOTE,*)'
CALL PTIME(TIMEOUT)
CALL DATIM(DATE1,TIME1)
WRITE(IOVOTE,9010)((TIMEOUT-TIMEIN)*3600.,TIME1
9010 FORMAT(' TIME USED:',F6.2,' SECONDS, COMPLETED AT',
&F6.2,' HOURS'))

```



```
10 IF(ALPHA .GT. 0.0 .AND. ALPHA .LT. 1.0) GO TO 15
      TERMINAL ERROR - ALPHA IS OUT OF
      RANGE
```

```

IER = 1.31
GO TO 9000
15 RN = NTRIAL
X = NX
RNMX = RN-X
PHAT = X/RN
AL = .5*ALPHA
IF (X .LE. 0.0) GO TO 20
CALL MDBET1(AL,X,RNMX+1.,PLOWER,JER)
IF(JER .NE. 0) GO TO 35
GO TO 25
20 PLOWER = 0.0
25 IF(X .GE. RN) GO TO 30
CALL MDBET1(AL,RNMX,X+1.,PUPPER,JER)
IF(JER .NE. 0) GO TO 35
PUPPER = 1.-PUPPER
GO TO 9005
30 PUPPER = 1.0
GO TO 9005
35 IER = 132
9000 CONTINUE
CALL UERTST(IER,6HBELBIN)
9005 RETURN
END

```

C	IMSL ROUTINE NAME	-	MOBETA
---	-------------------	---	--------

257

```

COMPUTER      - H1S/SINGLE
LATEST REVISION  - JULY 1, 1983
PURPOSE      - BETA PROBABILITY DISTRIBUTION FUNCTION
USAGE        - CALL MDBETA (X,A,B,P,IER)
ARGUMENTS    X  - VALUE TO WHICH FUNCTION IS TO BE INTEGRATED.
                A  - X MUST BE IN RANGE (0.1) INCLUSIVE.(INPUT)
                B  - INPUT (1ST) PARAMETER (MUST BE GREATER THAN 0)
                P  - INPUT (2ND) PARAMETER (MUST BE GREATER THAN 0)
                IER - OUTPUT PROBABILITY THAT A RANDOM VARIABLE
                    FROM A BETA DISTRIBUTION HAVING PARAMETERS
                    A AND B WILL BE LESS THAN OR EQUAL TO X.
                    - ERROR PARAMETER. (OUTPUT)
                    - TERMINAL ERROR
                    IER = 129 INDICATES X IS NOT IN RANGE (0.1)
                      INCLUSIVE
                    IER = 130 INDICATES A AND/OR B LESS THAN OR
                      EQUAL TO 0
PRECISION/HARDWARE  - SINGLE/ALL
REQD. IMSL ROUTINES - H32/MLGAMD=DLGAMA,UERTST,UGETIO
                  - H36,H48,H60/MLGAMA=ALGAMA,UERTST,UGETIO
NOTATION        - INFORMATION ON SPECIAL NOTATION AND
                  CONVENTIONS IS AVAILABLE IN THE MANUAL

```

INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
 - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
 - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
 APPLIED TO THIS CODE. NO OTHER WARRANTY,
 EXPRESSED OR IMPLIED, IS APPLICABLE.

SUBROUTINE MDBETA (X,A,B,P,IER)

X,A,B,P SPECIFICATIONS FOR ARGUMENTS
 AA,BB,TEMP SPECIFICATIONS FOR LOCAL VARIABLES
 PS,PX,Y,PI,DA,XINT,CNT,WM,XB,DB,C,EPS,EPS1,
 ALEPS,TOT,PQ,D4,ONE,ZERO
 INT1,IB MACHINE PRECISION
 EPS/1.D-7/
 SMALLEST POSITIVE NUMBER
 SAFELY REPRESENTABLE
 EPS1/1.D-38/
 NATURAL LOG OF EPS1
 ALEPS/-87.4982300/
 ONE/1.0D0/ZERO/0.0D0/
 CHECK RANGES OF THE ARGUMENTS
 FIRST EXECUTABLE STATEMENT

```

      Y = X
      IF ((X.LE. 1.0) .AND. (X.GE. 0.0)) GO TO 5
      IER = 129
      GO TO 9000
      5 IF ((A.GT. 0.0) .AND. (B.GT. 0.0)) GO TO 10
      IER = 130
      GO TO 9000
      10 IER = 0
      AA = A
      BB = B
      IF (X.GT. 0.5) GO TO 15
      INT1 = 0
      GO TO 20
  
```

SWITCH ARGUMENTS FOR MORE EFFICIENT
 USE OF THE POWER SERIES

```

      15 INT1 = 1
      TEMP = AA
      AA = BB
      BB = TEMP
      Y = ONE-Y
      20 IF (X.NE. 0. .AND. X.NE. 1.) GO TO 25
      SPECIAL CASE - X IS 0. OR 1.
      P = 0.E0
      GO TO 60
      25 TEMP = AINT(BB)
      PS = BB-TEMP
      IF (BB.EQ. TEMP) PS = ONE
      CA = AA
      CB = BB
      PX = DA*DLOG(Y)
      D4 = DLOG(DA)
  
```



```

C LATEST REVISION      - JUNE 1, 1980
C
C PURPOSE              - INVERSE BETA PROBABILITY DISTRIBUTION FUNCTION
C
C USAGE                - CALL MDETI (P,A,B,X,IER)
C
C ARGUMENTS            P      - INPUT PROBABILITY IN THE EXCLUSIVE RANGE
C                        (0,1)
C                        A      - INPUT FIRST PARAMETER OF THE BETA DISTRIBUTION
C                        B      - INPUT SECOND PARAMETER OF THE BETA DISTRIBUTION
C                        X      - OUTPUT VALUE SUCH THAT P IS THE PROBABILITY
C                              THAT A RANDOM VARIABLE DISTRIBUTED BETA(A,B)
C                              IS LESS THAN OR EQUAL TO X.
C                        IER     - ERROR PARAMETER. (OUTPUT)
C                              TERMINAL ERROR
C                              IER = 129 INDICATES THAT P WAS NOT IN THE
C                              EXCLUSIVE RANGE (0,1).
C                              IER = 130 INDICATES A AND/OR B LESS THAN
C                              OR EQUAL TO 0.
C                              IER = 131 INDICATES THAT THE VALUE X COULD
C                              NOT BE FOUND WITHIN 100 ITERATIONS OF
C                              NEWTON'S METHOD.
C
C PRECISION/HARDWARE   - SINGLE/ALL
C
C REQD. IMSL ROUTINES  - H32/MDDBETA,MLGAMD=DLGAMA,UERTST,UGETIO
C                      - H36,H48,H60/MDDBETA,MLGAMA=ALGAMA,UERTST,
C                      UGETIO
C
C NOTATION              - INFORMATION ON SPECIAL NOTATION AND
C                        CONVENTIONS IS AVAILABLE IN THE MANUAL
C                        INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C COPYRIGHT            - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C WARRANTY              - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                        APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                        EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----
C
C SUBROUTINE MDETI (P,A,B,X,IER)
C
C DATA                EPS=.0001/,SIG/1.E-4/,ITMAX/30/,ZERO/0.0/
C DATA                SMEXE/-.8E.0/
C                        FIRST EXECUTABLE STATEMENT
C
C IER = 0
C XL = AMINI(A,B)
C IF (XL.LE.0.0) GO TO 70
C IF (XL.LE.1.0) GO TO 20
C XR = AMAX1(A,B)
C IF (10.0*XL.LE.XR) GO TO 20
C IC = 0
C XL = 0.0
C XR = 1.0
C FXL = -P
C FXR = 1.0-P
C IF (FXL+FXR.GT.ZERO) GO TO 65

```

```

C
5 X = (XL*XR)*.5
  CALL MDBETA (X,A,B,P1,IER)
  IF (IER.NE.0) GO TO 20
  FCS = P1-P
  IF (FCS*FXL.GT.ZERO) GO TO 10
  XR = X
  FXR = FCS
  GO TO 15
10 XL = X
  FXL = FCS
15 XRMXL = XR-XL
  IF (XRMXL.LE.SIG.AND.ABS(FCS).LE.EPS) GO TO 9005
  IC = IC+1
  IF (IC.LE.ITMAX) GO TO 5
  ERROR RETURNED FROM MDBETA
  USE NEWTONS METHOD FOR SKEWED CASES
20 IF (P.LE.0.0.OR.P.GE.1.0) GO TO 65
  IF (P.GT..5) GO TO 25
  AA = A
  BB = B
  QQ = ALOG(P)
  GO TO 30
25 QQ = ALOG(1.0-P)
  AA = B
  BB = A
30 XI = AA/(AA+BB)
  DTEMP = ALGAMA(AA+BB) - ALGAMA(AA) - ALGAMA(BB)
  DTEMP = DTEMP-(AA+BB)*ALOG(AA+BB)+(AA-.5)*ALOG(AA)+(BB-.5)
  1*ALOG(BB)
  DTEMP = DTEMP+.5*ALOG(BB/AA)+AA*ALOG(1.0+BB/AA)+BB*ALOG(1.+AA/BB)
  DO 45 NC=1,200
    TEMP = ALOG(15.0+AA+BB)
    FN = 0.7*TEMP*TEMP*AMAX1(XI*(AA+BB)-AA,0.0)
    TEMP = AA+FN+FN
    AFN = AINT(FN)+1.0
    C = 1.0-(AA+BB)*XI/TEMP
    ZI = 2.0/(C+SQRT(C*C-4.0*FN*(FN-BB)*XI/(TEMP*TEMP)))
    AFN = AFN-1.0
    IF (AFN.LT..5) GO TO 40
    TEMP = AA+AFN+AFN
    ZI = (TEMP-2.0)*(TEMP-1.0-AFN*(AFN-BB)*XI*ZI/TEMP)
    TEMP = AA+AFN-1.0
    ZI = 1.0/(1.0-TEMP*(TEMP+BB)*XI/ZI)
    GO TO 35
  40 ZZ = ZI
    TEMP = ALOG(XI)
    IF (TEMP.LE.SMEXE) GO TO 50
    QX = DTEMP+AA*TEMP+BB*ALOG(1.-XI)+ALOG(ZZ)
    XC = (QQ-QX)*(1.-XI)*ZZ/AA
    XC = AMAX1(XC,-.99)
    TEMP = .5/XI-.5
    XC = AMIN1(XC,TEMP)
    XI = XI*(1.+XC)
    IF (ABS(XC).LT.SIG) GO TO 55
  45 CONTINUE
  IER = 131
  GO TO 9000
50 XI = 0.0
55 IF (P.GT..5) GO TO 60

```



```

10 IF (X.EQ.0) THEN
    ALGAMA = XINF
    GO TO 9000
END IF
10 MFLAG = .TRUE.
    T = -T
    R = AINT(T)
    XSIGN = 1.0
    IF (AMOD(R,2.0) .EQ. 0.0) XSIGN = -1.0
    R = T-R
    IF (R.NE.0.0) GO TO 15
    IER = 130
    ALGAMA = XINF
    GO TO 9000
END IF
15 R = PI/SIN(R*PI)*XSIGN
    T = T+1.0
    R = ALOG(ABS(R))
    ARGUMENT IS NOT A NEGATIVE INTEGER
EVALUATE APPROXIMATION FOR
LN(GAMMA(T)), T.GT. 0.0
0.0 .LT. T .LT. 0.5
0.5 .LE. T .LT. 1.5
1.5 .LE. T .LE. 4.0
4.0 .LT. T .LE. 12.0
20 IF (T.GT.12.0) GO TO 70
    IF (T.GT.4.0) GO TO 60
    IF (T.GE.1.5) GO TO 50
    IF (T.GE.0.5) GO TO 35
    B = -ALOG(T)
    A = T
    T = T+1.0
    IF (A.GE.EPS) GO TO 40
    ALGAMA = B
    GO TO 9005
END IF
35 TOP = T-0.5
    B = 0.0
    A = TOP-0.5
    40 TOP = P1(IEND1)*T+P1(IEND)
        DEN = T+Q1(IEND1)
        DO 45 J=1,IEND2
            TOP = TOP+T*P1(J)
            DEN = DEN+T*Q1(J)
        45 CONTINUE
        Y = (TOP/DEN)*A+B
        IF (MFLAG) Y = R-Y
        ALGAMA = Y
        GO TO 9005
END IF
50 B = T-1.0
    A = B-1.0
    TOP = P2(IEND1)*T+P2(IEND)
    DEN = T+Q2(IEND1)
    DO 55 J=1,IEND2
        TOP = TOP+T*P2(J)
        DEN = DEN+T*Q2(J)
    55 CONTINUE
    Y = (TOP/DEN)*A
    IF (MFLAG) Y = R-Y
    ALGAMA = Y
    GO TO 9005
END IF
60 TOP = P3(IEND1)*T+P3(IEND)

```

```

DEN = T*Q3(IEND1)
DO 65 J=1,IEND2
  TOP = TOP+T*P3(J)
  DEN = DEN+T*Q3(J)
65 CONTINUE
Y = TOP/DEN
IF (MFLAG) Y = R-Y
ALGAMA = Y
GO TO 9005

12.6 LET X .LT. BIG1

70 TOP = ALOG(T)
TOP = T*(TOP-1.0)-.5*TOP
T = 1.0/T
IF (T.LT.EPS) GO TO 75
B = T*T
TOP = (P4(3)*B+P4(2))*T+P4(1)+TOP
75 Y = TOP
IF (MFLAG) Y = R-Y
ALGAMA = Y
GO TO 9005
9000 CONTINUE
CALL UERTST(-IER,BHMLGAMA)
CALL UERTST(IER,BHMLGAMA)
9005 RETURN
END

IMSL ROUTINE NAME - UERTST
-----
C
C COMPUTER - HIS/SINGLE
C
C LATEST REVISION - JUNE 1, 1982
C
C PURPOSE - PRINT A MESSAGE REFLECTING AN ERROR CONDITION
C
C USAGE - CALL UERTST (IER,NAME)
C
C ARGUMENTS IER - ERROR PARAMETER (INPUT)
C IER = 1+J WHERE
C I = 128 IMPLIES TERMINAL ERROR MESSAGE,
C I = 64 IMPLIES WARNING WITH FIX MESSAGE,
C I = 32 IMPLIES WARNING MESSAGE.
C J = ERROR CODE RELEVANT TO CALLING
C ROUTINE.
C NAME - A CHARACTER STRING OF LENGTH SIX PROVIDING
C THE NAME OF THE CALLING ROUTINE. (INPUT)
C
C PRECISION/HARDWARE - SINGLE/ALL
C
C REQD. IMSL ROUTINES - UGETIO,USPKD
C
C NOTATION - INFORMATION ON SPECIAL NOTATION AND
C CONVENTIONS IS AVAILABLE IN THE MANUAL
C INTRODUCTION OR THROUGH IMSL ROUTINE UHELP
C
C REMARKS THE ERROR MESSAGE PRODUCED BY UERTST IS WRITTEN
C TO THE STANDARD OUTPUT UNIT. THE OUTPUT UNIT
C NUMBER CAN BE DETERMINED BY CALLING UGETIO AS
C FOLLOWS... CALL UGETIO(I,NIN,NOU).
C THE OUTPUT UNIT NUMBER CAN BE CHANGED BY CALLING

```

```

C
C
C      UGETIO AS FOLLOWS...
C      NIN = 0
C      NOUT = NEW OUTPUT UNIT NUMBER
C      CALL UGETIO(3,NIN,NOUT)
C      SEE THE UGETIO DOCUMENT FOR MORE DETAILS.
C
C      COPYRIGHT      - 1982 BY IMSL, INC. ALL RIGHTS RESERVED.
C      WARRANTY      - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                     APPLIED TO THIS CODE. NO OTHER WARRANTY,
C                     EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----
C
C      SUBROUTINE UERTST (IER,NAME)
C
C      INTEGER      IER
C      INTEGER      NAME(1)
C
C      INTEGER      I,IEQ,IEQDF,IOUNIT,LEVEL,LEVOLD,NAMEQ(6),
C      *            NAMSET(6),NAMUPK(6),NIN,NMTB
C      DATA        NAMESET/1HU,1HE,1HR,1HS,1HE,1HT/
C      DATA        NAMEQ/6*1H /
C      DATA        LEVEL/47,IEQDF/07,IEQ/1H=/
C
C      CALL USPXD (NAME,6,NAMUPK,NMTB)
C      UNPACK NAME INTO NAMUPK
C      FIRST EXECUTABLE STATEMENT
C      GET OUTPUT UNIT NUMBER
C      CHECK IER
C
C      IF (IER.GT.999) GO TO 25
C      IF (IER.LT.-32) GO TO 55
C      IF (IER.LE.128) GO TO 5
C      IF (LEVEL.LT.1) GO TO 30
C
C      IF (IEQDF.EQ.1) WRITE(IOUNIT,35) IER,NAMEQ,IEQ,NAMUPK
C      IF (IEQDF.EQ.0) WRITE(IOUNIT,35) IER,NAMUPK
C      GO TO 30
C
C      5 IF (IER.LE.64) GO TO 10
C      IF (LEVEL.LT.2) GO TO 30
C
C      IF (IEQDF.EQ.1) WRITE(IOUNIT,40) IER,NAMEQ,IEQ,NAMUPK
C      IF (IEQDF.EQ.0) WRITE(IOUNIT,40) IER,NAMUPK
C      GO TO 30
C
C      10 IF (IER.LE.32) GO TO 15
C
C      IF (LEVEL.LT.3) GO TO 30
C      IF (IEQDF.EQ.1) WRITE(IOUNIT,45) IER,NAMEQ,IEQ,NAMUPK
C      IF (IEQDF.EQ.0) WRITE(IOUNIT,45) IER,NAMUPK
C      GO TO 30
C
C      15 CONTINUE
C
C      DO 20 I=1,6
C      IF (NAMUPK(1).NE.NAMSET(1)) GO TO 25
C      20 CONTINUE
C      LEVOLD = LEVEL
C      LEVEL = IER
C      IER = LEVOLD
C      IF (LEVEL.LT.0) LEVEL = 4
C      IF (LEVEL.GT.4) LEVEL = 4

```

```

GO TO 30
25 CONTINUE
IF (LEVEL.LT.4) GO TO 30
PRINT NON-DEFINED MESSAGE
IF (IEQDF.EQ.1) WRITE(IUNIT,50) IER,NAMED,IEQ,NAMUPK
IF (IEQDF.EQ.0) WRITE(IUNIT,50) IER,NAMUPK
30 IEQDF = 0
RETURN
35 FORMAT(19H *** TERMINAL ERROR,10X,7H( IER = ,13,
1 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)
40 FORMAT(27H *** WARNING WITH FIX ERROR,2X,7H( IER = ,13,
1 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)
45 FORMAT(18H *** WARNING ERROR,11X,7H( IER = ,13,
1 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)
50 FORMAT(20H *** UNDEFINED ERROR,9X,7H( IER = ,15,
1 20H) FROM IMSL ROUTINE ,6A1,A1,6A1)

```

```

SAVE P FOR P = R CASE
P IS THE PAGE NAMUPK
R IS THE ROUTINE NAMUPK

```

```

55 IEQDF = 1
DO 60 I=1,D
60 NAMEQ(I) = NAMUPK(I)
65 RETURN
END

```

```

IMSL ROUTINE NAME - UGETIO

```

```

COMPUTER - HIS/SINGLE
LATEST REVISION - JUNE 1, 1981
PURPOSE - TO RETRIEVE CURRENT VALUES AND TO SET NEW
VALUES FOR INPUT AND OUTPUT UNIT
IDENTIFIERS.
USAGE - CALL UGETIO(IOPT,NIN,NOUT)
ARGUMENTS IOPT - OPTION PARAMETER. (INPUT)
IF IOPT=1, THE CURRENT INPUT AND OUTPUT
UNIT IDENTIFIER VALUES ARE RETURNED IN NIN
AND NOUT, RESPECTIVELY.
IF IOPT=2, THE INTERNAL VALUE OF NIN IS
RESET FOR SUBSEQUENT USE.
IF IOPT=3, THE INTERNAL VALUE OF NOUT IS
RESET FOR SUBSEQUENT USE.
NIN - INPUT UNIT IDENTIFIER.
NOUT - OUTPUT IF IOPT=1, INPUT IF IOPT=2,
OUTPUT UNIT IDENTIFIER.
OUTPUT IF IOPT=1, INPUT IF IOPT=3.
PRECISION/HARDWARE - SINGLE/ALL
REQD. IMSL ROUTINES NONE REQUIRED
NOTATION - INFORMATION ON SPECIAL NOTATION AND
CONVENTIONS IS AVAILABLE IN THE MANUAL
INTRODUCTION OR THROUGH IMSL ROUTINE UHELP

```



```

C REMARKS      EACH IMSL ROUTINE THAT PERFORMS INPUT AND/OR OUTPUT
C              OPERATIONS CALLS UGETIO TO OBTAIN THE CURRENT UNIT
C              IDENTIFIER VALUES. IF UGETIO IS CALLED WITH IOPT=2 OR
C              IOPT=3, NEW UNIT IDENTIFIER VALUES ARE ESTABLISHED.
C              SUBSEQUENT INPUT/OUTPUT IS PERFORMED ON THE NEW UNITS.
C
C COPYRIGHT    - 1978 BY IMSL, INC. ALL RIGHTS RESERVED.
C
C WARRANTY     - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C              APPLIED TO THIS CODE. NO OTHER WARRANTY,
C              EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----
C
C SUBROUTINE UGETIO(IOPT,NIN,NOUT)
C              SPECIFICATIONS FOR ARGUMENTS
C   INTEGER    IOPT,NIN,NOUT
C   INTEGER    NIND,NOUTD
C   DATA      NIND/5/,NOUTD/6/
C              FIRST EXECUTABLE STATEMENT
C
C   IF (IOPT.EQ.3) GO TO 10
C   IF (IOPT.EQ.2) GO TO 5
C   IF (IOPT.NE.1) GO TO 9005
C   NIN = NIND
C   NOUT = NOUTD
C   GO TO 9005
C 5  NIND = NIN
C   GO TO 9005
C 10 NOUTD = NOUT
C 9005 RETURN
C   END
C
C IMSL ROUTINE NAME      USPDK
C-----
C
C COMPUTER      - MVS/SINGLE
C
C LATEST REVISION - NOVEMBER 1, 1984
C
C PURPOSE       - NUCLEUS CALLED BY IMSL ROUTINES THAT HAVE
C               CHARACTER STRING ARGUMENTS
C
C USAGE         - CALL USPDK (PACKED,NCHARS,UNPAKD,NCHMTB)
C
C ARGUMENTS     PACKED - CHARACTER STRING TO BE UNPACKED.(INPUT)
C               NCHARS - LENGTH OF PACKED. (INPUT) SEE REMARKS.
C               UNPAKD - INTEGER ARRAY TO RECEIVE THE UNPACKED
C               REPRESENTATION OF THE STRING. (OUTPUT)
C               NCHMTB - NCHARS MINUS TRAILING BLANKS. (OUTPUT)
C
C PRECISION/HARDWARE - SINGLE/ALL
C
C REQD. IMSL ROUTINES - NONE
C
C REMARKS 1. USPDK UNPACKS A CHARACTER STRING INTO AN INTEGER ARRAY
C           IN (A1) FORMAT.
C           2. UP TO 129 CHARACTERS MAY BE USED. ANY IN EXCESS OF
C           THAT ARE IGNORED.

```

```

C  COPYRIGHT      - 1984 BY IMSL, INC.  ALL RIGHTS RESERVED.
C
C  WARRANTY      - IMSL WARRANTS ONLY THAT IMSL TESTING HAS BEEN
C                  APPLIED TO THIS CODE.  NO OTHER WARRANTY,
C                  EXPRESSED OR IMPLIED, IS APPLICABLE.
C
C-----
C  SUBROUTINE USPXD (PACKED,NCHARS,UNPAKD,NCHMTB)
C                  SPECIFICATIONS FOR ARGUMENTS
C  INTEGER      NC,NCHARS,NCHMTB
C
C  INTEGER      UNPAKD(1),IBLANK
C  CHARACTER*6   PACKED
C  DATA        IBLANK /1H /
C
C  NCHMTB = 0
C
C  IF(NCHARS.LE.0) RETURN
C
C  SET NC=NUMBER OF CHARS TO BE DECODED
C
C  NC = MIND (129,NCHARS)
C  READ(PACKED,150) (UNPAKD(I),I=1,NC)
C 150 FORMAT (129A1)
C
C  CHECK UNPAKD ARRAY AND SET NCHMTB
C  BASED ON TRAILING BLANKS FOUND
C
C  DO 200 N = 1,NC
C    NN = NC - N + 1
C    IF(UNPAKD(NN) .NE. IBLANK) GO TO 210
C 200 CONTINUE
C    NN = 0
C 210 NCHMTB = NN
C    RETURN
C  END

```

Appendix D - Sample GALACTIC Input

13. '9977B', 'ACG81', 'ACPH9', 'C1BDY', 'C2BDY', 'C3BDY', 'C4BDY',
 'C5BDY', 'C6BDY', 'C7BDY', 'C8BDY', '527540', '267410', 20, 'Y',
 9 2 3 6 9 10 11 14 15 16
 20
 9 00 01 20 21 02 03 04 05 27
 0 0
 0 0
 'RNU', 'RNU'
 0
 100100
 111100000010
 0
 ', ', 'CONTEST'
 0, 0
 0, 0
 0, 0
 ', '
 'CH15. DAT, 1', 'CH15. DAT, 1'
 0
 0
 0
 0

0	7	0.8240790	0.9276105	0.0000000E+00
1.0010000E-03	1.8702248E-04	0.0000000E+00	0.0000000E+00	0.2249505
0.0000000E+00	0.0000000E+00	2.0155622E-02	1.9414650E-02	0.2750238
0.0000000E+00	0.0000000E+00	4.4280000E-05	-4.0131032E-02	6.5958038E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0	22	21.73051	21.73053	0.0000000E+00
0.70076716	-4.3507316E-02	0.0000000E+00	0.0000000E+00	23.32464
0.0000000E+00	0.0000000E+00	0.8472973	0.1072113	-0.1456653
0.0000000E+00	0.0000000E+00	-3.6448685E-03	0.1075505	-3.3292517E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0	5	28.22051	28.22050	0.0000000E+00
1.021025	-5.6861322E-02	0.0000000E+00	0.0000000E+00	30.42640
0.0000000E+00	0.0000000E+00	1.114380	0.1269589	-0.3560690
0.0000000E+00	0.0000000E+00	-4.6405080E-03	0.1440598	-4.0388469E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0	23	0.2599925	0.2599437	0.0000000E+00
-2.4207475E-05	2.7965786E-04	0.0000000E+00	0.0000000E+00	3.4267720E-02
0.0000000E+00	0.0000000E+00	2.5798207E-04	3.3248086E-03	0.5580868
0.0000000E+00	0.0000000E+00	-4.8970216E-04	-1.1635082E-02	5.1717546E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0	22	0.5755203	0.5227754	0.0000000E+00
3.1011046E-03	-5.0478679E-04	0.0000000E+00	0.0000000E+00	0.3108462
0.0000000E+00	0.0000000E+00	-2.3448415E-02	3.0095249E-02	0.2768602
0.0000000E+00	0.0000000E+00	9.8913180E-05	1.6721459E-02	5.9865196E-03
6.2786712E-02	7.4157298E-02	-5.0872818E-02	-2.3260647E-03	0.0000000E+00
0	5	-1.049261	-1.061939	0.0000000E+00
6.3395800E-03	-8.1227126E-04	0.0000000E+00	0.0000000E+00	-0.1962710
0.0000000E+00	0.0000000E+00	-4.7657895E-02	-1.0935842E-02	1.6853672E-02
0.0000000E+00	0.0000000E+00	-1.3219503E-04	9.0055102E-03	1.2956944E-03
0.2257215	0.9997677	-5.3811787E-02	0.4225681	0.0000000E+00
0	23	-2.303845	-2.318939	0.0000000E+00
-0.7801617E-03	-1.4191014E-03	0.0000000E+00	0.0000000E+00	-0.9289371
0.0000000E+00	0.0000000E+00	1.4917985E-02	6.9554118E-03	0.1378282
0.0000000E+00	0.0000000E+00	3.4872988E-05	1.8711749E-02	6.7446232E-03
-0.941054E-02	5.4467764E-02	-5.5005637E-02	-3.5575449E-02	0.0000000E+00
0	5	3.9294232E-02	3.8033158E-02	0.0000000E+00
0.0000000E+00	-9.607284E-04	0.0000000E+00	0.0000000E+00	-4.6494271E-04
0.0000000E+00	0.0000000E+00	1.7119616E-02	-1.7073029E-03	-2.7587023E-02
0.0000000E+00	0.0000000E+00	-8.3054110E-05	-6.6154227E-03	-2.1943608E-03
-3.3176575E-02	1.9223335E-03	-1.3546035E-02	-6.8973869E-02	0.0000000E+00
0	23	-1.673259	-1.692602	0.0000000E+00
0.7008253E-03	4.1545777E-04	0.0000000E+00	0.0000000E+00	-0.7813047
0.0000000E+00	0.0000000E+00	1.3587150E-02	1.3430241E-02	-0.4653442
0.0000000E+00	0.0000000E+00	-2.6481457E-04	-2.3191137E-02	-5.4212278E-03
0.7072992E-03	-4.3115616E-02	1.8154845E-02	-7.1016550E-02	0.0000000E+00
0	23	-2.250376	-2.282809	0.0000000E+00
-1.0316902E-02	-1.0526507E-03	0.0000000E+00	0.0000000E+00	-0.9193538
0.0000000E+00	0.0000000E+00	8.1318943E-03	7.5071254E-03	0.1464850
0.0000000E+00	0.0000000E+00	6.4787317E-03	2.0933010E-02	7.4759726E-03
0.70084475	0.1827792	5.6158304E-03	-4.6592832E-02	0.0000000E+00
0	3	-0.2197392	-0.2190968	0.0000000E+00
0.0000000E+00	-2.7156108E-04	0.0000000E+00	0.0000000E+00	-1.1507622E-02
0.0000000E+00	0.0000000E+00	-3.6039092E-03	-1.4136495E-03	-0.4094222
0.0000000E+00	0.0000000E+00	3.5803340E-04	1.5038691E-02	-7.6856986E-02
0.4000000E+00	-0.1101012	-3.1344733E-03	-1.7132759E-03	0.0000000E+00
0	20	-0.5007082	-0.5007089	0.0000000E+00
0.0000000E+00	-6.2128601E-04	0.0000000E+00	0.0000000E+00	-5.1193018E-02
0.0000000E+00	0.0000000E+00	-1.1535180E-02	1.3952782E-02	-0.5413191
0.0000000E+00	0.0000000E+00	4.6965815E-04	-1.7259564E-02	2.5915673E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00

2	4	0	975251	-0	9924156	0	0000000E+00	
3	0111439E-03	8	2082272E-03	0	0000000E+00	0	0000000E+00	
4	0000000E+00	0	0000000E+00	3	7385977E-03	8	0581414E-03	
5	0000000E+00	0	0000000E+00	1	6620012E-04	1	9310748E-02	
6	0000000E+00	-9	9790573E-02	-1	8965095E-02	5	9065912E-02	
7	0000000E+00	20	-5	0485365E-02	-3	0487719E-02	0	0000000E+00
8	7185930E-04	-1	4394693E-04	0	0000000E+00	0	0000000E+00	
9	0000000E+00	0	0000000E+00	5	5211944E-03	5	2238707E-03	
10	0000000E+00	0	0000000E+00	2	5006749E-04	3	4267720E-02	
11	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
12	0000000E+00	4	-1	023433	-1	079199	0	0000000E+00
13	7380019E-04	5	1913581E-05	0	0000000E+00	0	0000000E+00	
14	0000000E+00	0	0000000E+00	5	1223373E-03	8	9555904E-03	
15	0000000E+00	0	0000000E+00	1	2253935E-04	1	5393170E-02	
16	0000000E+00	4	1293349E-02	-0	2750822	1	000090	
17	0000000E+00	4	1	2975-11E-02	1	2974638E-02	0	0000000E+00
18	2799810E-04	-7	6797725E-05	0	0000000E+00	0	0000000E+00	
19	0000000E+00	0	0000000E+00	4	8961971E-05	-4	701188E-03	
20	0000000E+00	0	0000000E+00	1	6226335E-04	-3	9245475E-02	
21	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
22	0000000E+00	2	0	5111591	0	5483046	0	0000000E+00
23	0000000E+00	4	0195975E-04	0	0000000E+00	0	0000000E+00	
24	0000000E+00	0	0000000E+00	5	2772149E-03	7	3158983E-03	
25	0000000E+00	0	0000000E+00	-4	2677630E-04	2	5514092E-02	
26	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
27	0000000E+00	3	0	2962304	0	2965649	0	0000000E+00
28	4311139E-04	3	6347702E-04	0	0000000E+00	0	0000000E+00	
29	0000000E+00	0	0000000E+00	4	4955690E-03	1	4898469E-03	
30	0000000E+00	0	0000000E+00	-3	7963823E-01	2	2022523E-02	
31	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
32	0000000E+00	4	0	3013018	0	3013037	0	0000000E+00
33	7932555E-04	3	8106649E-04	0	0000000E+00	0	0000000E+00	
34	0000000E+00	0	0000000E+00	5	0579237E-03	6	0029863E-04	
35	0000000E+00	0	0000000E+00	-5	4767961E-04	1	5726036E-03	
36	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
37	0000000E+00	10	-0	1e4745	-0	1634048	0	0000000E+00
38	0000000E+00	-3	1786819E-04	0	0000000E+00	0	0000000E+00	
39	0000000E+00	0	0000000E+00	5	8369591E-04	1	7095235E-03	
40	0000000E+00	0	0000000E+00	5	7006121E-04	1	0922260E-03	
41	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
42	0000000E+00	10	0	2472214	0	2473280	0	0000000E+00
43	0000000E+00	-1	7200478E-04	0	0000000E+00	0	0000000E+00	
44	0000000E+00	0	0000000E+00	1	4153545E-03	1	4601735E-02	
45	0000000E+00	0	0000000E+00	-2	8472473E-04	-6	8457223E-02	
46	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
47	0000000E+00	4	0	2082507	0	4994883	0	0000000E+00
48	0000000E+00	2	9601471E-04	0	0000000E+00	0	0000000E+00	
49	0000000E+00	0	0000000E+00	1	4153545E-03	1	4601735E-02	
50	0000000E+00	0	0000000E+00	-2	8472473E-04	-6	8457223E-02	
51	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
52	0000000E+00	2	9416232E-04	0	0000000E+00	0	0000000E+00	
53	0000000E+00	0	0000000E+00	-2	1023555E-02	1	2264006E-02	
54	0000000E+00	0	0000000E+00	-1	1092032E-04	3	1567797E-02	
55	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	
56	0000000E+00	3	0	121084	0	2082814	0	0000000E+00
57	0000000E+00	-1	5722604E-04	0	0000000E+00	0	0000000E+00	
58	0000000E+00	0	0000000E+00	-1	2461977E-02	1	4028946E-02	
59	0000000E+00	0	0000000E+00	-1	2358476E-04	-1	2897972E-02	
60	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00	

0	4	0	1750551	0	1994090	0	0000000E+00
4	1072376E-04	3	1156171E-04	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	3	1105785E-03	1	0997916E-04
0	0000000E+00	0	0000000E+00	5	4658845E-04	-4	5761033E-03
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
21	1	-0.171347	-0	1713012	0	0000000E+00	
0	0414009E-03	-1	8E73749E-04	0	0000000E+00	5	1234782E-02
0	0000000E+00	0	0000000E+00	7	7920735E-03	-0	5322345
0	0000000E+00	0	0000000E+00	3	0975265E-04	-7	9713175E-03
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
21	20	-7	2240622E-02	-4	242221E-02	0	0000000E+00
-1	702390E-03	-1	1756549E-04	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	1	1132955E-02	-3	2998407E-03
0	0000000E+00	0	0000000E+00	4	5274838E-04	1	4669186E-02
-9	4458557E-03	-6	8077490E-02	-9	0668641E-02	-6	3657482E-02
1	20	0.2794988	0	2794958	0	0000000E+00	
4	0013026E-03	4	1898688E-05	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	-1	0945333E-02	1	0942113E-02
0	0000000E+00	0	0000000E+00	-4	2033434E-04	1	1498610E-02
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
21	10	-0.432941	-0	2422430	0	0000000E+00	
2	3396760E-03	-3	8041692E-04	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	-7	3157344E-03	-1	4265542E-03
0	0000000E+00	0	0000000E+00	4	8753505E-04	2	7889528E-02
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
20	10	-0.4110599	-0	1805754	0	0000000E+00	
3	4529507E-03	-3	0252678E-04	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	-8	2077281E-03	-2	1495740E-03
0	0000000E+00	0	0000000E+00	4	5786555E-04	4	0426899E-02
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	20	23.07658	24	38633
1	075600	-5	5918535E-02	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	0	3945719	0	7504909
0	0000000E+00	0	0000000E+00	-1	3030957E-02	-0	6719638
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
21	0	-20	16562	-20	16585	0	0000000E+00
1	077251	6	9062322E-02	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	-0	7028303	-0	9178057
0	0000000E+00	0	0000000E+00	1	3084889E-02	1	110618
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
1	0672388E-04	5	2940213E-03	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	7	4593040E-03	1	1848936E-02
0	0000000E+00	0	0000000E+00	1	2015631E-04	4	2882677E-02
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
1	0	0.232655	0	2568044	0	0000000E+00	
-1	4013191E-03	3	8358339E-04	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	7	0010051E-03	-7	3218596E-04
0	0000000E+00	0	0000000E+00	-3	7345841E-04	2	6801020E-02
0	0000000E+00	-2	2170709E-02	-9	0022112E-04	-1	8296598E-02
0	0000000E+00	0	0000000E+00	0	0000000E+00	0	0000000E+00
0	0000000E+00	-2	9241765E-04	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	-0	3781627E-03	4	4831430E-04
0	0000000E+00	0	0000000E+00	2	5259574E-04	6	3622579E-02
7	4070095E-03	1	8480514E-03	2	6367782E-02	3	0446211E-02
2	7	3268388E-02	7	3268388E-02	0	0000000E+00	
1	0907032E-02	3	2631406E-04	0	0000000E+00	0	0000000E+00
0	0000000E+00	0	0000000E+00	1	0815877E-02	-5	4139541E-03
0	0000000E+00	0	0000000E+00	-8	5408581E-05	1	9421577E-02
0	0000000E+00	-5	6983831E-03	9	2840791E-03	-3	6627650E-03

1	0.12599E-03	1	0.27343E-04	32	0.39442E90	0.4119532	0.0000000E+00	0.0000000E+00
2	0.000000E+00	0	0.000000E+00	1	0.8797958E-02	6.3992417E-03	0.3532918	0.7171111E-02
3	0.000000E+00	0	0.000000E+00	2	4.819610E-04	-5.3887633E-03	7.6162145E-02	0.000000E+00
4	0.2500779	-8	4.306617E-03	4	4.833183E-02	-7.9101473E-02	0.000000E+00	0.000000E+00
5	0.151174E-02	2	0.8103578	5	0.7408233	0.000000E+00	0.000000E+00	0.000000E+00
6	0.000000E+00	0	0.000000E+00	6	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
7	0.000000E+00	0	0.000000E+00	7	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
8	0.000000E+00	0	0.000000E+00	8	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
9	0.4725924E-03	-0.2443150	-4.7987498E-02	1	1.481559E-02	1.481559E-02	1.2465645E-02	1.2465645E-02
10	0.000000E+00	0	0.000000E+00	2	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
11	0.000000E+00	0	0.000000E+00	3	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
12	0.000000E+00	0	0.000000E+00	4	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
13	0.000000E+00	0	0.000000E+00	5	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
14	0.000000E+00	0	0.000000E+00	6	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
15	0.000000E+00	0	0.000000E+00	7	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
16	0.000000E+00	0	0.000000E+00	8	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
17	0.000000E+00	0	0.000000E+00	9	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
18	0.000000E+00	0	0.000000E+00	10	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
19	0.000000E+00	0	0.000000E+00	11	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
20	0.000000E+00	0	0.000000E+00	12	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
21	0.000000E+00	0	0.000000E+00	13	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
22	0.000000E+00	0	0.000000E+00	14	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
23	0.000000E+00	0	0.000000E+00	15	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
24	0.000000E+00	0	0.000000E+00	16	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
25	0.000000E+00	0	0.000000E+00	17	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
26	0.000000E+00	0	0.000000E+00	18	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
27	0.000000E+00	0	0.000000E+00	19	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
28	0.000000E+00	0	0.000000E+00	20	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
29	0.000000E+00	0	0.000000E+00	21	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
30	0.000000E+00	0	0.000000E+00	22	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
31	0.000000E+00	0	0.000000E+00	23	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
32	0.000000E+00	0	0.000000E+00	24	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
33	0.0000							

AD-A195 699

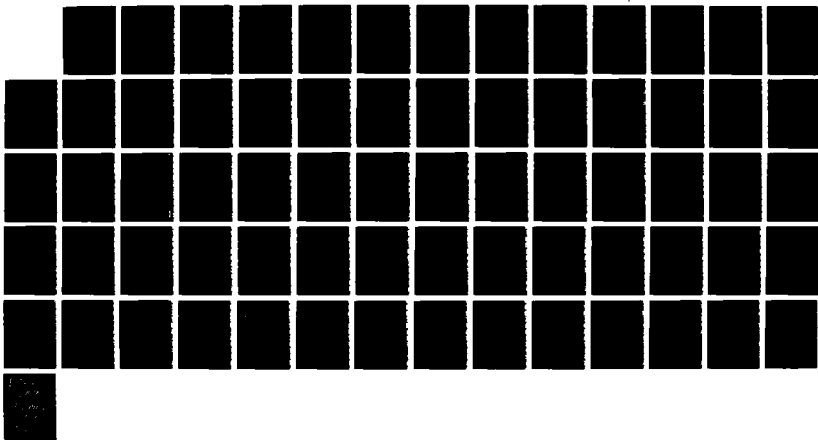
EMPIRICAL FLUTTER PREDICTION METHOD(U) GE AIRCRAFT
ENGINES CINCINNATI OH ADVANCED TECHNOLOGY OPERATION
J K CASEY 85 MAR 88 R07AEG AFMAL-TR-87-2007

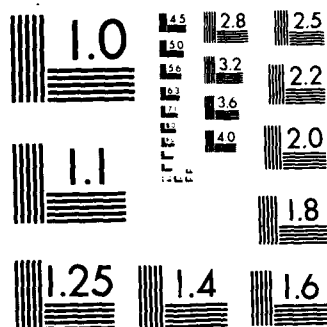
4/4

UNCLASSIFIED F33615-84-C-2457

F/G 20/4

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Appendix E - Sample GALACTIC Output

ENTER N (LE 20) THEN ID OF N SETS TO BE EXCLUDED 2

ENTER N (LE 20) THEN NAME N POINTS TO BE EXCLUDED

ENTER N, THEN N PAIRS OF IDS TO BE LABELED AS THE FIRST

ORIGINAL POINT NUMBERS FOR INCLUDED POINTS

82	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115
116	117	118	119	120	121	122	123	124	125	126	127	128	129	130
131	132	133	134	135	136	137	138	139	140	141	142	143	144	145
146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190
191	192	193	194	195	196	197	198	199	200	201	202	203	204	205
206	207	208	209	210	211	212	213	214	215	216	217	218	219	220
221	222	223	224	225	226	227	228	229	230	231	232	233	234	235
236	237	238	239	240	241	242	243	244	245	246	247	248	249	250
251	252	253	254	255	256	257	258	259	260	261	262	263	264	265
266	267	268	269	270	271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290	291	292	293	294	295
296	297	298	299	300	301	302	303	304	305	306	307	308	309	310
311	312	313	314	315	316	317	318	319	320	321	322	323	324	325
326	327	328	329	330	331	332	333	334	335	336	337	338	339	340
341	342	343	344	345	346	347	348	349	350	351	352	353	354	355
356	357	358	359	360	361	362	363	364	365	366	367	368	369	370
371	372	373	374	375	376	377	378	379	380	381	382	383	384	385
386	387	388	389	390	391	392	393	394	395	396	397	398	399	400

USUAL TYPICAL SCALE OPTIONS(MIN,MAX,AVER,SIGMA,MID,RANGE,SPECIAL)

ENTER COUNT OF BONDED POINTS(12J FORCED INTO SAME CLUSTER)

ENTER 0 OR 1 TO NOT WRITE OR WRITE FOLLOWING OUTPUT OPTIONS.

FOR ALL DATA: SMH, NTR, NPT, BAX, BTR, BPT

FOR CLUSTER: IPT, MEM, USU, HPL, BAX, PRJ, BTR, CGR, COR, EST, HPP, HPO

FOR CLUSTER: EDS, SDS, STP, FPI, MEM, CDD, BAX, PRJ, BTR, CGR, COR, SUR

SET NUMBER	SET ID	NO. OF POINTS
1	10	201
2	7	31
3	22	13
4	5	2
5	23	4

NUMBER OF EACH VARIABLE FOR THE	MIN	MID	MAX	RANGE	AVER	SIGMA
1	0.2680E+02	0.4730E+02	0.6850E+02	0.4170E+02	0.4937E+02	0.1565E+02
2	0.5240E+03	0.7260E+03	0.9180E+03	0.3930E+03	0.7113E+03	0.1425E+03
3	0.9740E+00	0.1380E+01	0.1760E+01	0.7030E+00	0.1233E+01	0.1883E+00
4	0.4190E+01	0.1312E+02	0.2205E+02	0.1750E+02	0.1008E+02	0.4723E+01
5	0.8710E+01	0.3240E+01	0.1480E+02	0.2351E+02	0.4503E+01	0.3805E+01
6	0.3470E+00	0.6760E+00	0.1398E+01	0.1036E+01	0.7126E+00	0.1930E+00
7	0.3070E+02	0.1030E+01	0.1650E+04	0.1827E+04	0.8720E+03	0.2144E+03
8	0.1350E+01	0.4038E+01	0.7340E+01	0.5975E+01	0.4190E+01	0.1213E+01
9	0.5100E+00	0.9200E+01	0.9930E+01	0.9420E+01	0.1483E+01	0.6674E+00

CLUSTER 0: BATTLE CONTAINING SET 1

0.13 7.134 -0.211 0.419 0.151 -0.023 -0.045 -0.202 0.902

-0.019 0.007 -0.005 -0.167 0.127 0.335 0.435 0.321 0.239
 0.302 -0.397 -0.019 0.377 -0.211 0.445 0.228 -0.037 -0.367
 -0.490 -0.483 -0.122 -0.122 -0.062 0.235 0.076 0.160 0.306
 -0.125 -0.006 -0.443 -0.126 -0.207 0.000 -0.032 -0.191 -0.007
 0.118 0.084 0.161 -0.116 -0.051 -0.002 0.025 -0.036 0.108
 -0.103 0.072 0.080 0.063 0.027 0.038 0.066 -0.232 0.001
 0.044 -0.031 0.022 -0.142 0.078 0.104 -0.067 -0.071 -0.002
 -0.041 0.038 -0.004 0.036 -0.018 0.053 -0.079 0.015 0.005
 LENGTH OF THE SEMIAXES
 1.163 1.153 0.890 0.854 0.561 0.446 0.294 0.235 0.113

SEMIAXES OF BOTTLE CONTAINING SET 2
 -0.077 0.302 -0.385 -0.123 -0.145 0.446 0.477 0.218 0.089
 -0.061 -0.541 0.158 0.168 -0.208 0.295 0.101 0.117 0.031
 0.302 -0.076 -0.262 0.185 0.050 0.043 -0.030 -0.072 -0.021
 0.160 0.125 0.191 0.093 -0.075 -0.016 0.049 0.122 0.032
 0.093 -0.008 0.037 -0.071 0.002 0.053 -0.011 -0.035 -0.008
 0.007 -0.011 -0.017 -0.022 -0.025 -0.041 0.014 -0.017 0.002
 0.013 0.025 -0.004 0.020 -0.035 0.032 -0.046 -0.019 -0.011
 0.002 -0.004 -0.008 -0.010 -0.005 -0.002 -0.016 0.024 0.002
 0.000 0.000 0.000 0.000 0.000 0.000 -0.001 -0.001 0.005
 LENGTH OF THE SEMIAXES
 0.875 0.713 0.518 0.338 0.125 0.086 0.078 0.032 0.006

SEMIAXES OF BOTTLE CONTAINING SET 3
 -0.140 0.148 -0.731 0.005 0.255 0.268 0.262 0.071 0.061
 -0.163 -0.645 -0.236 0.106 -0.201 -0.026 -0.130 -0.025 -0.011
 0.361 0.043 -0.165 0.073 -0.182 -0.061 -0.058 0.111 0.020
 -0.161 0.120 -0.068 -0.133 -0.192 -0.089 -0.077 0.091 -0.012
 0.002 -0.002 0.023 -0.021 0.055 -0.007 -0.011 0.134 0.011
 0.017 -0.004 -0.023 -0.044 0.051 -0.047 -0.057 -0.032 -0.006
 0.009 0.005 -0.002 0.019 0.054 -0.006 -0.005 -0.032
 -0.004 0.003 0.000 0.010 0.001 -0.002 -0.007 0.000 0.009
 0.000 0.000 0.000 0.000 0.000 0.000 -0.001 0.000 0.000
 LENGTH OF THE SEMIAXES
 0.688 0.754 0.482 0.349 0.152 0.107 0.039 0.016 0.002

SEMIAXES OF BOTTLE CONTAINING SET 4
 -0.002 0.000 0.000 -0.006 -0.022 -0.020 -0.022 -0.017 -0.006
 LENGTH OF THE SEMIAXES
 0.042

SEMIAXES OF BOTTLE CONTAINING SET 5
 0.197 0.098 -0.061 0.003 0.022 0.030 0.048 0.354 0.062
 0.243 0.172 -0.005 -0.128 0.057 0.077 0.103 0.071 0.021
 -0.065 -0.154 -0.019 0.015 -0.001 0.018 -0.006 0.072 0.015
 LENGTH OF THE SEMIAXES
 0.411 0.344 0.186

KNOWN VERTEX POINTS OF EACH SET
 SET NUMBER 1 (27 OF 201 POINTS)
 1 15 48 49 51 113 118 124 131 141 150 156 169 177 178
 217 193 195 206 217 218 222 224 225 231 234 250
 SET NUMBER 2 (22 OF 31 POINTS)
 10 12 20 21 25 75 77 78 85 87 94 95 96 98 99
 103 110 120 122 173 174 175
 SET NUMBER 3 (13 OF 13 POINTS)
 62 107 116 124 137 139 161 162 164 165 166 168 176
 SET NUMBER 4 (2 OF 2 POINTS)
 128 129

SETS 1 4 DISJOINT IN SPACE OF FIRST 4 AXES
 SETS 1 5 DISJOINT IN SPACE OF FIRST 5 AXES
 SETS 2 3 DISJOINT IN SPACE OF FIRST 6 AXES
 SETS 2 4 DISJOINT IN SPACE OF FIRST 5 AXES
 SETS 2 5 DISJOINT IN SPACE OF FIRST 3 AXES
 SETS 3 4 DISJOINT IN SPACE OF FIRST 4 AXES
 SETS 3 5 DISJOINT IN SPACE OF FIRST 5 AXES
 SETS 4 5 DISJOINT IN SPACE OF FIRST 2 AXES

HYPERPLANES FOR SET DISCRIMINATION, IN TERMS OF TRANSFORMED VARIABLES

ZX1LP HAS DONE 100 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 200 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 300 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 400 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 500 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 600 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 700 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 800 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 900 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1000 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1100 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1200 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1300 ITERATIONS OF ALLOWED 2000
 HYPERPLANES FOR SETS 1(-0.1480E+00 LE RHS) 2(-0.1209E+00 GE RHS)
 RHS IS - 0.100E+00*(1) - 0.196E+00*(2) + 0.280E+00*(3)
 + 0.215E+00*(5) - 0.939E-01*(6) - 0.203E+00*(7) + 0.574E+00*(8)
 - 0.281E+00*(9) + 0.574E+00*(10)

ZX1LP HAS DONE 100 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 200 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 300 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 400 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 500 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 600 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 700 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 800 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 900 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1000 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1100 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1200 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1300 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1400 ITERATIONS OF ALLOWED 2000
 HYPERPLANES FOR SETS 1(-0.1137E-01 LE RHS) 2(-0.3383E-02 GE RHS)
 RHS IS + 0.147E+00*(1) - 0.656E-01*(2) + 0.709E-02*(3)
 + 0.102E+00*(5) + 0.166E-02*(6) + 0.165E+00*(7) + 0.126E+00*(8)
 + 0.528E-01*(9) + 0.958E+00*(10)

ZX1LP HAS DONE 100 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 200 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 300 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 400 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 500 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 600 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 700 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 800 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 900 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1000 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1100 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1200 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1300 ITERATIONS OF ALLOWED 2000
 ZX1LP HAS DONE 1400 ITERATIONS OF ALLOWED 2000

ZXILP HAS DONE	1500 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	1600 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	1700 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	1800 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	1900 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	2000 ITERATIONS OF ALLOWED	2000
DISCRIM INP =	132 FOR SETS	4
TRYING OPPOSITE	CASE AS HYPERPLANES NOT SOLVABLE FOR SETS	1, 4
ZXILP HAS DONE	100 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	200 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	300 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	400 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	500 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	600 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	700 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	800 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	900 ITERATIONS OF ALLOWED	2000
TRYING SPACE OF	5 VARIABLES	
ZXILP HAS DONE	100 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	200 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	300 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	400 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	500 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	600 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	700 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	800 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	900 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	1000 ITERATIONS OF ALLOWED	2000
DISCRIM INP =	70 FOR SETS	4
TRYING OPPOSITE	CASE AS HYPERPLANES NOT SOLVABLE FOR SETS	1, 4
ZXILP HAS DONE	100 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	200 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	300 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	400 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	500 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	600 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	700 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	800 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	900 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	1000 ITERATIONS OF ALLOWED	2000
TRYING SPACE OF	6 VARIABLES	
ZXILP HAS DONE	100 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	200 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	300 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	400 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	500 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	600 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	700 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	800 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	900 ITERATIONS OF ALLOWED	2000
ZXILP HAS DONE	1000 ITERATIONS OF ALLOWED	2000

2X1LP HAS DONE	2000 ITERATIONS OF ALLOWED	1	4	2000
DISCRIMINER =	137 FOR SETS	1	4	
TRYING UPSITE CASE AS HYPERPLANES NOT SOLVABLE FOR SETS				
2X1LP HAS DONE	100 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	200 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	300 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	400 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	500 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	600 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	700 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	800 ITERATIONS OF ALLOWED	2000		
TRYING SPACE OF	9 VARIABLES			
2X1LP HAS DONE	100 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	200 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	300 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	400 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	500 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	600 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	700 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	800 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	900 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1000 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1100 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1200 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1300 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1400 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1500 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1600 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1700 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1800 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1900 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	2000 ITERATIONS OF ALLOWED	2000		
DISCRIMINER =	137 FOR SETS	1	4	
TRYING UPSITE CASE AS HYPERPLANES NOT SOLVABLE FOR SETS				
2X1LP HAS DONE	100 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	200 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	300 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	400 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	500 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	600 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	700 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	800 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	900 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1000 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1100 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1200 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1300 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1400 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1500 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1600 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1700 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1800 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	1900 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	2000 ITERATIONS OF ALLOWED	2000		
DISCRIMINER =	137 FOR SETS	1	4	
TRYING UPSITE CASE SEPARATION CASE SOLVABLE FOR SETS				
2X1LP HAS DONE	100 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	200 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	300 ITERATIONS OF ALLOWED	2000		
2X1LP HAS DONE	400 ITERATIONS OF ALLOWED	2000		

FAIL HAS DONE	500 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	600 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	700 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	800 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	900 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1000 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1100 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1200 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1300 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1400 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1500 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1600 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1700 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1800 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	1900 ITERATIONS OF ALLOWED	2000
FAIL HAS DONE	2000 ITERATIONS OF ALLOWED	2000
132 FOR SETS	5	5
CASE AS HYPERPLANES NOT SOLVABLE FOR SETS	1.	5
100 ITERATIONS OF ALLOWED	2000	
200 ITERATIONS OF ALLOWED	2000	
300 ITERATIONS OF ALLOWED	2000	
400 ITERATIONS OF ALLOWED	2000	
500 ITERATIONS OF ALLOWED	2000	
600 ITERATIONS OF ALLOWED	2000	
700 ITERATIONS OF ALLOWED	2000	
800 ITERATIONS OF ALLOWED	2000	
900 ITERATIONS OF ALLOWED	2000	
1000 ITERATIONS OF ALLOWED	2000	
1100 ITERATIONS OF ALLOWED	2000	
6 VARIABLES		
130 ITERATIONS OF ALLOWED	2000	
200 ITERATIONS OF ALLOWED	2000	
300 ITERATIONS OF ALLOWED	2000	
400 ITERATIONS OF ALLOWED	2000	
500 ITERATIONS OF ALLOWED	2000	
600 ITERATIONS OF ALLOWED	2000	
700 ITERATIONS OF ALLOWED	2000	
800 ITERATIONS OF ALLOWED	2000	
900 ITERATIONS OF ALLOWED	2000	
1000 ITERATIONS OF ALLOWED	2000	
1100 ITERATIONS OF ALLOWED	2000	
1200 ITERATIONS OF ALLOWED	2000	
1300 ITERATIONS OF ALLOWED	2000	
1400 ITERATIONS OF ALLOWED	2000	
1500 ITERATIONS OF ALLOWED	2000	
1600 ITERATIONS OF ALLOWED	2000	
1700 ITERATIONS OF ALLOWED	2000	
1800 ITERATIONS OF ALLOWED	2000	
1900 ITERATIONS OF ALLOWED	2000	
2000 ITERATIONS OF ALLOWED	2000	
137 FOR SETS	5	5
CASE AS HYPERPLANES NOT SOLVABLE FOR SETS	1.	5
100 ITERATIONS OF ALLOWED	2000	
200 ITERATIONS OF ALLOWED	2000	
300 ITERATIONS OF ALLOWED	2000	
400 ITERATIONS OF ALLOWED	2000	
500 ITERATIONS OF ALLOWED	2000	
600 ITERATIONS OF ALLOWED	2000	
700 ITERATIONS OF ALLOWED	2000	
800 ITERATIONS OF ALLOWED	2000	

[illegible]

HYPERPLANES FOR SETS 2(-0.4052E+00 LE RHS) 5(-0.4812E+00 GE RHS)
 RHS IS + 0.497E+00*(1) - 0.847E+00*(2) - 0.189E+00*(3)
 HYPERPLANES FOR SETS 3(0.5907E-01 LE RHS) 4(-0.2640E-01 GE RHS)
 RHS IS - 0.569E+00*(1) + 0.271E+00*(2) + 0.655E+00*(3)
 + 0.417E+00*(5)
 HYPERPLANES FOR SETS 3(-0.6306E-01 LE RHS) 5(-0.8351E-01 GE RHS)
 RHS IS + 0.505E+00*(1) - 0.272E+00*(2) - 0.793E+00*(3)
 + 0.189E+00*(5) + 0.880E-01*(5)
 HYPERPLANES FOR SETS 4(-0.6260E+00 LE RHS) 5(-0.8486E+00 GE RHS)
 RHS IS - 0.321E+00*(1) - 0.947E+00*(2)

OVERLAP OF SETS 1 2 2 = 0.2711E-01
 OVERLAP OF SETS 1 2 3 = 0.7905E-02
 SEPARATION BETWEEN SETS 1 2 4 = 0.1740E-04
 SEPARATION BETWEEN SETS 1 2 5 = -0.5812E+00
 SEPARATION BETWEEN SETS 2 2 3 = 0.2520E-01
 SEPARATION BETWEEN SETS 2 2 4 = 0.3388E-03
 SEPARATION BETWEEN SETS 2 2 5 = -0.3916E-02
 SEPARATION BETWEEN SETS 3 2 4 = 0.5347E-01
 SEPARATION BETWEEN SETS 3 2 5 = 0.2045E-01
 SEPARATION BETWEEN SETS 4 2 5 = 0.2226E+00

SAMPLE PLANES APPLIED TO TRANSFORMED SAMPLE FILES, LESS EXCLUSIONS

POINT SET	DISTANCE(RHS)	TO POINT	NORMAL TO PLANES (E=ERROR, D=BAND)	1	2	3	4	5
110 1 0.1138 -0.0024	9.9999	-0.11236	0.0435	0.1592	0.2089	0.0764		
210 1-0 0.0323 -0.4021	9.9999	-0.10103	0.0508	0.1241	0.1609	0.0072		
310 1-0 0.0553 -0.0073	9.9999	-0.10128	0.0503	0.1247	0.1639	0.0069		
410 1-0 0.0402 -0.0053	9.9999	-0.10352	0.0338	0.1576	0.1528	0.0458		
510 1-0 0.0432 -0.0046	9.9999	-0.10494	0.0320	0.1605	0.1502	0.0434		
610 1 0.0002 0.0020	9.9999	-0.10861	0.0127	0.2018	0.1336	0.0824		
710 1-0 0.0043 -0.0020	9.9999	-0.10826	0.0168	0.1911	0.1408	0.0736		
810 1-0 0.0000 0.0094	9.9999	-0.10145	0.0432	0.1440	0.1460	0.0231		
910 1-0 0.0333 -0.0025	9.9999	-0.10110	0.0478	0.1320	0.1549	0.0132		
1010 1-0 0.0663 -0.0067	9.9999	-0.10108	0.0453	0.1374	0.1522	0.0181		
1110 1-0 0.0326 -0.0027	9.9999	-0.10478	0.0308	0.1636	0.1395	0.0482		
1210 1-0 0.0306 -0.0073	9.9999	-0.10390	0.0275	0.1752	0.1352	0.0537		
1310 1-0 0.0122 -0.0080	9.9999	-0.10285	0.0124	0.2070	0.1234	0.0840		
1410 1-0 0.0112 -0.0091	9.9999	-0.10748	0.0119	0.2076	0.1250	0.0854		
1510 1-0 0.0223 -0.0131	9.9999	-0.09980	0.0423	0.1503	0.177	0.0270		
1610 1-0 0.0225 -0.0119	9.9999	-0.10028	0.0411	0.1478	0.1373	0.0244		
1710 1-0 0.0121 -0.0132	9.9999	-0.09776	0.0384	0.1611	0.1289	0.0357		
1810 1-0 0.0096 -0.0151								

1810	1-0.0621	0.0100	9.9999	-0.10320	0.0257	0.1877	0.1196	0.0617
	-0.1014	-0.5156						
1910	1-0.0231	0.0090	9.9999	-0.10540	0.0098	0.2182	0.1108	0.0930
	-0.1290	-0.5029						
2010	1-0.0944	0.0137	9.9999	-0.10008	0.0337	0.1790	0.1138	0.0903
	-0.1069	-0.5260						
2110	1-0.0605	0.0154	9.9999	-0.10206	0.0184	0.2140	0.0973	0.0832
	-0.1428	-0.5315						
2210	1-0.0624	0.0132	9.9999	-0.10240	0.0212	0.2021	0.1074	0.0732
	-0.1246	-0.5245						
2310	1-0.0390	0.0137	9.9999	-0.10336	0.0115	0.2259	0.0964	0.0972
	-0.1514	-0.5166						
2410	1-0.1220	-0.0014	9.9999	-0.11068	0.0433	0.1571	0.2090	0.0786
	0.0868	-0.3840						
2510	1-0.1463	-0.0012	2.9954	-0.07358	0.0500	0.0308	0.3191	0.0419
	0.1410	0.2669						
2610	1-0.0455	0.0153	4.4564	-0.07536	0.0895	0.0239	0.2402	0.0145
	0.0451	0.1788						
2710	1-0.0634	0.0182	4.4241	-0.07448	0.0950	0.0209	0.2344	0.0100
	0.0397	0.1714						
2810	1-0.0244	0.0160	4.3755	-0.07768	0.0790	0.0535	0.2223	0.0418
	0.0120	0.1690						
2910	1-0.0154	0.0147	4.1232	-0.08126	0.0618	0.0938	0.2046	0.0800
	-0.0278	0.1642						
3010	1-0.0118	0.0143	4.1489	-0.08110	0.0637	0.0884	0.2082	0.0751
	-0.0215	0.1659						
3110	1-0.0016	0.0187	4.1788	-0.08086	0.0631	0.1061	0.1875	0.0894
	-0.0538	0.1525						
3210	1-0.0668	0.0167	4.1920	-0.08150	0.0629	0.1008	0.1941	0.0853
	-0.0428	0.1558						
3310	1-0.0253	0.0178	4.3543	-0.07848	0.0763	0.0686	0.2086	0.0547
	-0.0117	0.1612						
3410	1-0.0808	0.0223	4.7930	-0.07370	0.0977	0.0294	0.2191	0.0155
	0.0175	0.1583						
3510	1-0.0902	0.0250	4.9125	-0.07389	0.0971	0.0489	0.1966	0.0313
	-0.0159	0.1410						
3610	1-0.0524	0.0247	4.4201	-0.07780	0.0818	0.0828	0.1826	0.0643
	-0.0164	0.1304						
3710	1-0.0091	0.0244	4.3531	-0.08080	0.0638	0.1264	0.1648	0.1056
	-0.0894	0.1346						
3810	1-0.0039	0.0231	4.2400	-0.09070	0.0623	0.1242	0.1682	0.1044
	-0.0841	0.1339						
3910	1-0.0528	0.0286	4.6370	-0.07760	0.0793	0.1027	0.1654	0.0814
	-0.0764	0.1270						
4010	1-0.0967	0.0310	4.9457	-0.07420	0.0765	0.0677	0.1773	0.0472
	-0.0467	0.1282						
4110	1-0.0511	0.0505	4.3557	-0.07288	0.0715	0.1861	0.0869	0.1553
	0.1495	0.0825						
4210	1-0.1080	0.0555	4.5844	-0.06878	0.1184	0.1070	0.1129	0.0819
	-0.1235	0.0891						
4310	1-0.0942	0.0495	4.7459	-0.07030	0.0594	0.1400	0.1091	0.1108
	-0.1545	0.0883						
4410	1-0.0501	0.0421	4.9016	-0.07230	0.0926	0.1122	0.1347	0.0859
	-0.1139	0.1016						
4510	1-0.0626	0.0414	4.6859	-0.07358	0.0850	0.1264	0.1288	0.1001
	-0.1369	0.1029						
4610	1-0.0372	0.0422	4.7397	-0.07270	0.0853	0.1210	0.1315	0.0948
	-0.1217	0.1032						
4710	1-0.0813	0.0414	4.8361	-0.07300	0.0856	0.1250	0.1341	0.0981
	-0.1242	0.1046						

4010	1	0	1468	-0.0006	2.5140	-0.07208	0.0526	0.0236	0.3264	0.0360
			0.1517	0.2726						
4910	1	0	2287	-0.0114E	0.1369	-0.02688	-0.0004	0.1411	0.0533	0.1233
			0.0510	-0.1600						
50	7	2	-0.12108	0.0112	9.9999	-0.0084	0.0703	-0.0033	0.0348	-0.0443
			0.0658	-0.2473						
5110	1	-0	0.0095	0.0014	9.9999	-0.01238	0.0628	0.0237	0.0605	-0.0083
			0.0380	-0.2076						
5210	1	0	0014	0.0013	5.9999	-0.01518	0.0487	0.0595	0.0351	0.0212
			0.0358	-0.2259						
5310	1	0	0000	0.0028	9.9999	-0.01790	0.0369	0.0963	-0.0012	0.0516
			-0.0225	-0.2532						
5410	1	-0	0.0370	0.0116	9.9999	-0.01578	0.0406	0.1161	-0.0385	0.0628
			-0.0772	-0.2892						
5510	1	-0	0.0192	0.0030	9.9999	-0.01408	0.0535	0.0512	0.0255	0.0125
			0.0367	-0.2309						
5610	1	0	0114	0.0004	9.9999	-0.01558	0.0494	0.0552	0.0359	0.0203
			0.0500	-0.2241						
5710	1	-0	0.0675	0.0148	9.9999	-0.01328	0.0502	0.1022	-0.0388	0.0477
			-0.0737	-0.2935						
58	7	2	-0.12498	0.0262	9.9999	-0.0108	0.0661	0.1031	-0.0679	0.0431
			-0.1051	-0.3230						
5910	1	-0	0.0674	0.0600	9.9999	0.00278	0.0381	0.2542	-0.1763	0.1797
			-0.3022	-0.3757						
60	7	2	-0.12220	0.0544	9.9999	0.0126	0.0755	0.2853	-0.2397	0.2105
			-0.3612	-0.4120						
61	7	2	-0.1208E	0.0297	9.9999	-0.0057	0.0628	0.1126	-0.0649	0.0499
			-0.1189	-0.3187						
6220	0	0	0.0323	-0.0034E	9.9999	-0.0500	0.0231	0.1494	-0.0303	0.1060
			-0.0631	-0.2616						
6310	1	0	0449	-0.0015	9.9999	-0.01732	0.0540	0.0397	0.0698	0.0135
			0.0791	-0.1880						
6410	1	-0	0.0415	0.0034	5.1900	-0.00458	0.0576	-0.0234	-0.1544	-0.0133
			0.0969	-0.2420						
65	7	2	-0.14658	0.0027	5.4586	-0.0148	0.0763	-0.0393	-0.2020	-0.0321
			0.0735	-0.3392						
6610	1	-0	0.0308	0.0000	2.9381	-0.01568	0.0472	0.0186	-0.1973	0.0254
			0.0401	-0.3191						
67	7	2	-0.1649	0.0101	3.3804	-0.0282	0.0939	0.0235	-0.2630	0.0226
			-0.0215	-0.3771						
6810	1	-0	1.0398	0.0071	3.5607	-0.02540	0.0823	0.0197	-0.2473	0.0185
			-0.0106	-0.3675						
6910	1	-0	0.0251	-0.0621	0.0067	-0.03120	0.0348	0.0873	-0.2565	0.0856
			-0.0550	-0.3595						
7010	1	0	1176	0.0067	3.6013	-0.03980	0.0527	0.0963	-0.3048	0.0863
			-0.1104	-0.4046						
7110	1	-0	1.081E	0.0137	3.4771	-0.03380	0.0749	0.1030	-0.3241	0.0915
			-0.1331	-0.4152						
7210	1	-0	0.0989	0.0007	2.4208	-0.03780	0.0524	0.1207	-0.3191	0.1091
			-0.1387	-0.4054						
7310	1	-0	1.0448	0.0146	3.5123	-0.03090	0.0597	0.1265	-0.3455	0.1120
			0.0662	-0.4304						
7410	1	0	1.0648	0.0147	2.5199	-0.03880	0.0713	0.1251	-0.3424	0.1107
			0.1641	-0.4271						
75	7	2	0.1983	0.0241	2.9207	-0.0481	0.1040	0.1228	-0.3751	0.1119
			0.1719	-0.4490						
7610	1	-0	1.0278	0.0212	3.4314	-0.04040	0.0572	0.1749	-0.3914	0.1551
			0.2740	-0.4009						
77	7	2	0.1898	0.0326	2.7659	-0.0504	0.0926	0.1786	-0.4258	0.1620
			-0.2454	-0.4793						

71	7	2-0	1746	0.0292	3.0000	-0.0491	0.0049	0.1789	-0.4160	0.1604
			-0.2462	-0.4753						
72	7	2-0	1531	0.0006	3.4049	-0.0190	0.0932	-0.0189	-0.2232	-0.0146
			0.0424	-0.3544						
80	7	2-0	1733	0.0117	3.6504	-0.0276	0.0963	0.0247	-0.2587	0.0213
			-0.0284	-0.3752						
8110	1	0	0667	-0.00468	2.5581	-0.02038	0.0118	0.0801	-0.2108	0.0867
			0.0009	-0.3206						
8210	1-0	0	0429	-0.0032	9.9999	-0.03868	0.0339	0.1028	-0.2610	0.0586
			-0.0210	-0.6535						
83	7	2-0	1641	0.0074	9.9999	-0.0415	0.0722	0.1023	-0.3204	0.0509
			-0.0763	-0.7061						
8410	1-0	0	1207	0.0002	9.9999	-0.04318	0.0509	0.1004	-0.2960	0.0485
			-0.0550	-0.6525						
8510	1-0	0	12098	0.0008	9.9999	-0.04820	0.0467	0.1300	-0.3238	0.0743
			-0.1027	-0.7106						
86	7	2-0	1876	0.0161	9.9999	-0.0712	0.0688	0.2015	-0.4274	0.1355
			-0.2289	-0.7863						
87	7	2-0	1789	0.0306	9.9999	-0.0676	0.0628	0.2683	-0.4794	0.1952
			-0.3217	-0.8124						
88	7	2-0	1849	0.0173	9.9999	-0.0689	0.0671	0.2020	-0.4164	0.1342
			-0.2296	-0.7785						
8910	1-0	0	14358	0.0063	9.9999	-0.04998	0.0485	0.1562	-0.3504	0.0936
			-0.1935	-0.7312						
90	7	2-0	1921	0.0138	9.9999	-0.0594	0.0788	0.1528	-0.3766	0.0932
			-0.1582	-0.7482						
91	7	2-0	1668	0.0079	9.9999	-0.0425	0.0708	0.1111	-0.3276	0.0575
			-0.0908	-0.7131						
9210	1-0	0	0391	-0.00468	9.9999	-0.04858	0.0218	0.1554	-0.3062	0.1036
			-0.0989	-0.6829						
93	7	2-0	1570	0.0059	9.9999	-0.0371	0.0725	0.0780	-0.2913	0.0292
			-0.0391	-0.6859						
94	7	2-0	1602	0.0060	9.9999	-0.0344	0.0790	0.0562	-0.2744	0.0111
			-0.0058	-0.6752						
95	7	2-0	1772	0.0056	9.9999	-0.0293	0.0986	0.0023	-0.2351	-0.0355
			0.0693	-0.6526						
96	7	2-0	1819	0.0122	9.9999	-0.0221	0.1096	-0.0393	-0.2008	-0.0706
			0.1306	-0.6279						
9710	1-0	0	0485	0.0036	4.8697	-0.01263	0.0413	-0.0195	-0.3603	-0.0032
			0.0953	-0.5734						
98	7	2-0	1697	0.0045	4.9479	-0.0278	0.0830	-0.0393	-0.4172	-0.0249
			0.0730	-0.6269						
99	7	2-0	12378	0.0029	5.0859	-0.0212	0.0628	-0.0354	-0.4073	-0.0202
			0.0641	-0.6242						
10010	1-0	0	0358	0.0066	4.4610	-0.01795	0.0318	0.0019	-0.3921	0.0184
			0.0691	-0.5939						
101	7	2-0	1714	0.0010	4.8081	-0.0367	0.0789	-0.0130	-0.4450	-0.0015
			0.0347	-0.6457						
10210	1-0	0	0407	-0.0025	4.5666	-0.02592	0.0299	0.0223	-0.4112	0.0357
			0.0367	-0.6085						
103	7	2-0	1932	-0.0009	4.8931	-0.0506	0.0944	0.0127	-0.4797	0.0209
			-0.0074	-0.6713						
10410	1-0	0	0339	-0.00548	4.2950	-0.0310E	0.0241	0.0390	-0.4336	0.0522
			0.0166	-0.6222						
10510	1-0	0	0451	-0.01028	4.4156	-0.04945	0.0205	0.0930	-0.4795	0.0983
			-0.0638	-0.6518						
10610	1-0	0	0391	-0.00738	4.5810	-0.03908	0.0234	0.0641	-0.4435	0.0724
			-0.0190	-0.6261						
10732	2	0	0336	-0.01058	3.8507	-0.0496	0.0609	0.1100	-0.4573	0.1204
			-0.0466	-0.6247						

10010	1-0	0.0226	0.0055	4.2435	-0.007308	0.0326	-0.0207	-0.3713	0.0003
		0.1058	-0.5774						
10910	1-0	0.0622	0.0071	9.9999	-0.03458	0.0420	0.0519	-0.4501	0.0204
		0.0725	-0.9787						
11010	2-0	0.1781	0.0048	9.9999	-0.0510	0.0794	0.0351	-0.4789	-0.0021
		0.0471	-1.0079						
11110	1-0	0.0572	0.0017	9.9999	-0.04408	0.0339	0.0795	-0.4693	0.0444
		0.0337	-0.9883						
11210	1-0	0.0606	-0.0026	9.9999	-0.05248	0.0302	0.1011	-0.4809	0.0617
		0.0012	-0.9946						
11310	1-0	0.0551	-0.00548	9.9999	-0.05828	0.0231	0.1208	-0.5102	0.0807
		-0.0263	-1.0154						
11410	1-0	0.0676	-0.00838	9.9999	-0.06708	0.0227	0.1428	-0.5274	0.0985
		-0.0645	-1.0220						
11510	1-0	0.0701	-0.0114E	9.9999	-0.07478	0.0168	0.1663	-0.5441	0.1173
		-0.0971	-1.0408						
11622	3-0	0.0445	-0.01008	9.9999	-0.0969	0.0153	0.1699	-0.5410	0.1311
		-0.0630	-1.0286						
11710	1	0.0264	0.0006	3.9658	-0.02378	0.0122	0.0328	-0.3763	0.0521
		0.0624	-0.5665						
11810	1	0.0544	-0.0011	3.5471	-0.02648	0.0066	0.0548	-0.3947	0.0752
		0.0437	-0.5750						
11910	1	0.0621	0.0060	3.5056	-0.00958	0.0061	0.0144	-0.3518	0.0400
		0.1035	-0.5434						
12010	1	0.0238	0.0127	9.9999	-0.02898	0.0020	0.0724	-0.4184	0.0490
		0.0928	-0.9317						
12110	1	0.0399	0.0074	9.9999	-0.03718	0.0106	0.1016	-0.4378	0.0755
		0.0587	-0.9442						
12210	1	0.0382	0.0004	9.9999	-0.03858	0.0269	0.0932	-0.2214	0.0648
		0.0493	-0.6070						
12310	1	0.0333	-0.0023	9.9999	-0.04668	0.0228	0.1165	-0.2316	0.0830
		0.0127	-0.6111						
12410	1-0	0.1195	0.0127	4.7529	-0.00318	0.0585	-0.0969	-0.3391	-0.0723
		0.1758	-0.5675						
12510	1-0	0.1059	0.0086	4.8243	-0.01028	0.0421	-0.0723	-0.3393	-0.0510
		0.1489	-0.5612						
12610	1-0	0.1181	0.0051	4.8676	-0.01868	0.0411	-0.0550	-0.3659	-0.0371
		0.1201	-0.5834						
12710	1-0	0.1134	0.0005	4.8311	-0.02918	0.0536	-0.0226	-0.3914	-0.0075
		0.0758	-0.5986						
12810	4-0	0.1477	0.0024	5.2030E-0	0.0296	0.0726	-0.0396	-0.3933	-0.0264
		0.0784	-0.6064						
12910	4-0	0.1845	0.0019	5.5521E-0	0.0358	0.0637	-0.0396	-0.4077	-0.0294
		0.0629	-0.6260						
13010	1-0	0.1458	-0.01138	4.6147	-0.07108	0.0434	0.0970	-0.5223	0.0978
		-0.1055	-0.6870						
13110	1-0	0.1279	-0.00978	4.6644	-0.05928	0.0428	0.0624	-0.4809	0.0677
		-0.0498	-0.6587						
13210	1-0	0.1248	-0.00658	4.7141	-0.04898	0.0464	0.0318	-0.4516	0.0406
		-0.0057	-0.6395						
13310	1-0	0.1124	-0.00498	4.6615	-0.04298	0.0130	0.0193	-0.4350	0.0304
		0.0171	-0.6272						
13420	3	0.0000	-0.0146	2.6634	-0.0816	-0.0007	0.1001	-0.4812	0.1379
		-0.0433	-0.6302						
13510	1-0	0.14158	0.0129	9.9999	-0.03298	0.0729	-0.0100	-0.4128	-0.0387
		0.1313	-0.9572						
13610	1-0	0.13048	0.0076	9.9999	-0.04198	0.0618	0.0200	-0.4381	-0.0112
		0.0925	-0.9717						
13710	1-0	0.12518	0.0036	9.9999	-0.04688	0.0530	0.0426	-0.4604	0.0086
		0.0601	-0.9570						

13910	1-0.12308-0.0011	9.9999	-0.03708	0.0405	0.0676	-0.4635	0.0293
	0.0204 -1.0040						
13910	1-0.12498-0.00448	9.9999	-0.06398	0.0416	0.0920	-0.4996	0.0508
	-0.0104 -1.0133						
14010	1-0.14028-0.00750	9.9999	-0.07280	0.0412	0.1117	-0.5244	0.0664
	-0.0449 -1.0355						
14110	1-0.14168-0.01028	9.9999	-0.07958	0.0361	0.1335	-0.5463	0.0855
	-0.0768 -1.0505						
14210	1-0.14810-0.0028	9.9999	-0.05293	0.0598	0.0418	-0.4668	0.0056
	0.0533 -0.9981						
14310	1-0.12648-0.0085	4.9472	-0.02108	0.0696	-0.0686	-0.3657	-0.0476
	0.1386 -0.5904						
14410	1-0.0750-0.0049	4.4798	-0.02480	0.0482	-0.0311	-0.3838	-0.0099
	0.1080 -0.5953						
14510	1-0.0757-0.0015	4.5406	-0.03328	0.0457	-0.0085	-0.3916	0.0090
	0.0767 -0.5955						
14610	1-0.0724-0.0025	4.4473	-0.04318	0.0392	0.0204	-0.4222	0.0353
	0.0379 -0.6172						
14710	1-0.0836-0.00518	4.3639	-0.05030	0.0389	0.0344	-0.4448	0.0476
	0.0125 -0.6331						
14810	1-0.0491-0.00678	4.4627	-0.05065	0.0364	0.0609	-0.4667	0.0703
	-0.0255 -0.6485						
14910	1-0.0942-0.01140	4.4409	-0.07028	0.0341	0.0878	-0.4924	0.0939
	-0.0647 -0.6651						
15010	1-0.0886-0.0203	9.9999	-0.02548	0.0676	-0.0099	-0.3939	-0.0314
	0.1611 -0.9332						
15110	1-0.0806-0.0122	9.9999	-0.03740	0.0549	0.0274	-0.4198	0.0003
	0.1097 -0.9505						
15210	1-0.0917-0.0052	9.9999	-0.05088	0.0485	0.0553	-0.4516	0.0233
	0.0636 -0.9756						
15310	1-0.1037-0.0015	9.9999	-0.06510	0.0439	0.0856	-0.4823	0.0479
	0.0143 -0.9996						
15410	1-0.1071-0.00628	9.9999	-0.07622	0.0372	0.1160	-0.5133	0.0743
	-0.0308 -1.0214						
15510	1-0.1115-0.01148	9.9999	-0.08946	0.0307	0.1526	-0.5467	0.1057
	-0.0648 -1.0445						
15610	1-0.0239-0.0074	4.5609	0.05850	0.0354	0.1543	-0.2701	0.1274
	-0.1973 -0.3587						
15722	3-0.0800-0.0190	9.9999	0.1137	-0.0707	0.1200	-0.4732	0.0754
	-0.0631 -0.8362						
15820	5-0.0534-0.0195	9.9999	0.11580	-0.0701	0.1213	-0.48258	0.0717
	-0.0835 -0.8464						
15922	3-0.0499-0.0218	9.9999	0.0804	-0.0815	0.1360	-0.7205	0.0964
	-0.0631 -1.1954						
16020	5-0.0160-0.0260	9.9999	0.08548	-0.0805	0.1379	-0.7341	0.0931
	-0.0861 -1.2112						
16112	3-0.0612-0.0197	1.0106	0.1251	-0.0723	0.0500	-0.6454	0.0614
	-0.0280 -0.8056						
16212	3-0.0438-0.0240	1.1812	0.1211	-0.0756	0.0668	-0.6680	0.0738
	-0.0631 -0.8231						
16312	5-0.0052-0.0253	1.1707	0.11618	-0.0601	0.0896	-0.7058	0.0921
	-0.1128 -0.8486						
16402	3-0.0681-0.01040	-0.9617	0.0145	0.0339	0.0462	-0.4638	0.0591
	-0.0175 -0.5447						
16502	3-0.1100-0.01010	9.9999	-0.0055	0.0376	0.0930	-0.5170	0.0591
	-0.0502 -0.8762						
16602	3-0.1205-0.0124	0.5100	-0.0001	0.0227	0.0359	-0.7033	0.0591
	-0.0600 -0.8984						
16720	5-0.0269-0.0209	9.9999	-0.07708	0.0497	0.1437	-0.8561	0.1034
	-0.1310 -1.3216						

22810	1	0.1564	-0.01088	9.9999	-0.02458	-0.0830	0.2011	0.2094	0.0708
		-0.0792	-0.4357						
22910	1	0.1215	-0.00908	9.9999	-0.02230	-0.0690	0.1717	0.2201	0.0426
		-0.0524	-0.4342						
23010	1	0.0619	-0.00618	9.9999	-0.01820	-0.0460	0.1249	0.2357	-0.0036
		-0.0137	-0.4349						
23110	1	-0.0022	-0.0014	9.9999	-0.01508	-0.0160	0.0746	0.2508	-0.0511
		0.0320	-0.4333						
23210	1	0.0134	-0.0002	9.9999	-0.00928	-0.0339	0.1003	0.2375	-0.0317
		-0.0082	-0.4443						
23310	1	0.0716	0.0269	9.9999	0.02658	-0.0909	0.2443	0.1372	0.0873
		-0.2252	-0.5061						
23410	1	0.1513	0.0393	9.9999	0.03888	-0.1283	0.3381	0.0877	0.1737
		-0.3326	-0.5240						
23510	1	0.0948	0.0183	9.9999	0.02110	-0.1036	0.2437	0.1511	0.0867
		-0.2147	-0.4979						
23610	1	0.0487	0.0225	9.9999	0.02088	-0.0771	0.2142	0.1542	0.0610
		-0.1864	-0.4976						
23710	1	0.1182	0.0193	9.9999	0.02182	-0.1140	0.2683	0.1394	0.1099
		-0.2406	-0.5012						
23810	1	0.0534	-0.0029	9.9999	-0.01100	-0.0517	0.1299	0.2299	-0.0036
		-0.0346	-0.4427						
23910	1	0.1346	-0.00698	9.9999	-0.01548	-0.0869	0.1996	0.2038	0.0633
		-0.0924	-0.4479						
24010	1	0.1023	-0.00568	9.9999	-0.01408	-0.0732	0.1727	0.2136	0.0374
		-0.0738	-0.4467						
24110	1	0.1134	-0.0032	9.9999	-0.00850	-0.0251	0.1963	0.1990	0.0556
		-0.1113	-0.4578						
24210	1	0.0694	-0.0013	9.9999	-0.00590	-0.0684	0.1599	0.2124	0.0201
		-0.0793	-0.4567						
24310	1	0.0063	0.0034	9.9999	-0.00458	-0.0359	0.1113	0.2355	-0.0245
		-0.0319	-0.4557						
24410	1	0.0129	0.0057	9.9999	0.00110	-0.0476	0.1308	0.2127	-0.0100
		-0.0658	-0.4645						
24510	1	0.0424	0.0041	9.9999	0.00180	-0.0639	0.1523	0.2081	0.0090
		-0.0074	-0.4658						
24610	1	0.0960	0.0012	9.9999	-0.00090	-0.0871	0.1957	0.1936	0.0507
		-0.1263	-0.4671						
24710	1	0.1168	0.0003	9.9999	0.00668	-0.1031	0.2342	0.1691	0.0834
		-0.1802	-0.4807						
24810	1	0.0405	0.0086	9.9999	0.00868	-0.0702	0.1673	0.1951	0.0195
		-0.1168	-0.4764						
24910	1	0.1364	-0.0114	9.9999	-0.03245	-0.0560	0.1810	0.2320	0.0411
		-0.0142	-0.4164						
25010	1	0.1860	-0.00830	9.9999	-0.06048	-0.0565	0.1441	0.2498	0.0573
		0.0532	-0.3617						
25110	1	0.1747	-0.00768	9.9999	-0.05938	-0.0035	0.1376	0.2514	0.0507
		0.0976	-0.3624						
L TOTALS	5	5	5	15	0	0	0	0	0
		0	0						
R TOTALS	33	44		0	205	0	0	1	0
	0	0							

MINIMUM OF HYPERPLANE RIS FOR LEFT SETS

-0.1481 -0.0114 -2.1884 -0.1106 0.0628 -0.0393 -0.4797 0.0591

MAXIMUM OF HYPERPLANE RIS FOR RIGHT SETS

-0.1208 -0.0033 5.5561 0.1161 0.0376 -0.0396 -0.4825 -0.0264

	LEFT	INO	RIGHTNO.	LEFTOLD	HIGHTOLD	LEFTNEW	RIGHTNEW
1	0	6278E-01	0	3750E-01	0	6278E-01	0
2	3	-0	3930E-01	0	3944E-01	0	3798E-01
3	4	0	5907E-01	0	2640E-01	0	2640E-01
4	0	6306E-01	0	8351E-01	0	6306E-01	0
5	-0	6260E+00	0	8486E+00	0	6260E+00	0

NUMBER OF SET TO WHICH POINT BELONGS (-MEANS REDUNDANT POINT)

[illegible]

NAME: REVISED SAMPLE & POPULATION FILES (IN SAMPLE FORMAT) (, IF NONE)

153411

ENTER N(LE 14) THEN IS OF N SETS TO BE EXCLUDED (-1 FOR ALL NEW SETS)

ENTERED N(LE 20) THEN NAME N POPULATION POINTS TO BE EXCLUDED

ENTER 4. THEN N PAIRS OF IDS TO BE LABELED AS THE FIRST

○

SAMPLE PLANES APPLIED TO TRANSFORMED POP FILES, LESS NEW EXCLUSIONS									
POINT SET		DISTANCE(RHS)		TO POINT,		NORMAL TO PLANES		(E=ERROR, B=BAND)	
NO.	NO.	1	2	1	2	1	2	1	2
1	1	1	2	1	4	1	5	2	3
2	1	2	1	2	1	4	1	5	2
3	1	2	1	2	1	4	1	5	2
4	1	2	1	2	1	4	1	5	2
5	1	2	1	2	1	4	1	5	2
6	1	2	1	2	1	4	1	5	2
7	1	2	1	2	1	4	1	5	2
8	1	2	1	2	1	4	1	5	2
9	1	2	1	2	1	4	1	5	2
10	1	2	1	2	1	4	1	5	2
11	1	2	1	2	1	4	1	5	2
12	1	2	1	2	1	4	1	5	2
13	1	2	1	2	1	4	1	5	2
14	1	2	1	2	1	4	1	5	2
15	1	2	1	2	1	4	1	5	2
16	1	2	1	2	1	4	1	5	2
17	1	2	1	2	1	4	1	5	2
18	1	2	1	2	1	4	1	5	2
19	1	2	1	2	1	4	1	5	2
20	1	2	1	2	1	4	1	5	2
21	1	2	1	2	1	4	1	5	2
22	1	2	1	2	1	4	1	5	2
23	1	2	1	2	1	4	1	5	2
24	1	2	1	2	1	4	1	5	2
25	1	2	1	2	1	4	1	5	2
26	1	2	1	2	1	4	1	5	2
27	1	2	1	2	1	4	1	5	2
28	1	2	1	2	1	4	1	5	2
29	1	2	1	2	1	4	1	5	2
30	1	2	1	2	1	4	1	5	2
31	1	2	1	2	1	4	1	5	2
32	1	2	1	2	1	4	1	5	2
33	1	2	1	2	1	4	1	5	2
34	1	2	1	2	1	4	1	5	2
35	1	2	1	2	1	4	1	5	2
36	1	2	1	2	1	4	1	5	2
37	1	2	1	2	1	4	1	5	2
38	1	2	1	2	1	4	1	5	2
39	1	2	1	2	1	4	1	5	2
40	1	2	1	2	1	4	1	5	2
41	1	2	1	2	1	4	1	5	2
42	1	2	1	2	1	4	1	5	2
43	1	2	1	2	1	4	1	5	2
44	1	2	1	2	1	4	1	5	2
45	1	2	1	2	1	4	1	5	2
46	1	2	1	2	1	4	1	5	2
47	1	2	1	2	1	4	1	5	2
48	1	2	1	2	1	4	1	5	2
49	1	2	1	2	1	4	1	5	2
50	1	2	1	2	1	4	1	5	2
51									

100 600 94974 0 04733 1 21941-0 32006 0 33776 0 05793-0 11583 6 3374=

0 17475--0.07492

0.0079 0.0086 0.0093 0.0100 0.0107 0.0114 0.0121 0.0128 0.0135 0.0142 0.0149 0.0156 0.0163 0.0170 0.0177 0.0184 0.0191 0.0198 0.0205 0.0212 0.0219 0.0226 0.0233 0.0240 0.0247 0.0254 0.0261 0.0268 0.0275 0.0282 0.0289 0.0296 0.0303 0.0310 0.0317 0.0324 0.0331 0.0338 0.0345 0.0352 0.0359 0.0366 0.0373 0.0380 0.0387 0.0394 0.0401 0.0408 0.0415 0.0422 0.0429 0.0436 0.0443 0.0450 0.0457 0.0464 0.0471 0.0478 0.0485 0.0492 0.0499 0.0506 0.0513 0.0520 0.0527 0.0534 0.0541 0.0548 0.0555 0.0562 0.0569 0.0576 0.0583 0.0590 0.0597 0.0604 0.0611 0.0618 0.0625 0.0632 0.0639 0.0646 0.0653 0.0660 0.0667 0.0674 0.0681 0.0688 0.0695 0.0702 0.0709 0.0716 0.0723 0.0730 0.0737 0.0744 0.0751 0.0758 0.0765 0.0772 0.0779 0.0786 0.0793 0.0800 0.0807 0.0814 0.0821 0.0828 0.0835 0.0842 0.0849 0.0856 0.0863 0.0870 0.0877 0.0884 0.0891 0.0898 0.0905 0.0912 0.0919 0.0926 0.0933 0.0940 0.0947 0.0954 0.0961 0.0968 0.0975 0.0982 0.0989 0.0996 0.1003 0.1010 0.1017 0.1024 0.1031 0.1038 0.1045 0.1052 0.1059 0.1066 0.1073 0.1080 0.1087 0.1094 0.1101 0.1108 0.1115 0.1122 0.1129 0.1136 0.1143 0.1150 0.1157 0.1164 0.1171 0.1178 0.1185 0.1192 0.1199 0.1206 0.1213 0.1220 0.1227 0.1234 0.1241 0.1248 0.1255 0.1262 0.1269 0.1276 0.1283 0.1290 0.1297 0.1304 0.1311 0.1318 0.1325 0.1332 0.1339 0.1346 0.1353 0.1360 0.1367 0.1374 0.1381 0.1388 0.1395 0.1402 0.1409 0.1416 0.1423 0.1430 0.1437 0.1444 0.1451 0.1458 0.1465 0.1472 0.1479 0.1486 0.1493 0.1500 0.1507 0.1514 0.1521 0.1528 0.1535 0.1542 0.1549 0.1556 0.1563 0.1570 0.1577 0.1584 0.1591 0.1598 0.1605 0.1612 0.1619 0.1626 0.1633 0.1640 0.1647 0.1654 0.1661 0.1668 0.1675 0.1682 0.1689 0.1696 0.1703 0.1710 0.1717 0.1724 0.1731 0.1738 0.1745 0.1752 0.1759 0.1766 0.1773 0.1780 0.1787 0.1794 0.1801 0.1808 0.1815 0.1822 0.1829 0.1836 0.1843 0.1850 0.1857 0.1864 0.1871 0.1878 0.1885 0.1892 0.1899 0.1906 0.1913 0.1920 0.1927 0.1934 0.1941 0.1948 0.1955 0.1962 0.1969 0.1976 0.1983 0.1990 0.1997 0.2004 0.2011 0.2018 0.2025 0.2032 0.2039 0.2046 0.2053 0.2060 0.2067 0.2074 0.2081 0.2088 0.2095 0.2102 0.2109 0.2116 0.2123 0.2130 0.2137 0.2144 0.2151 0.2158 0.2165 0.2172 0.2179 0.2186 0.2193 0.2200 0.2207 0.2214 0.2221 0.2228 0.2235 0.2242 0.2249 0.2256 0.2263 0.2270 0.2277 0.2284 0.2291 0.2298 0.2305 0.2312 0.2319 0.2326 0.2333 0.2340 0.2347 0.2354 0.2361 0.2368 0.2375 0.2382 0.2389 0.2396 0.2403 0.2410 0.2417 0.2424 0.2431 0.2438 0.2445 0.2452 0.2459 0.2466 0.2473 0.2480 0.2487 0.2494 0.2501 0.2508 0.2515 0.2522 0.2529 0.2536 0.2543 0.2550 0.2557 0.2564 0.2571 0.2578 0.2585 0.2592 0.2599 0.2606 0.2613 0.2620 0.2627 0.2634 0.2641 0.2648 0.2655 0.2662 0.2669 0.2676 0.2683 0.2690 0.2697 0.2704 0.2711 0.2718 0.2725 0.2732 0.2739 0.2746 0.2753 0.2760 0.2767 0.2774 0.2781 0.2788 0.2795 0.2802 0.2809 0.2816 0.2823 0.2830 0.2837 0.2844 0.2851 0.2858 0.2865 0.2872 0.2879 0.2886 0.2893 0.2900 0.2907 0.2914 0.2921 0.2928 0.2935 0.2942 0.2949 0.2956 0.2963 0.2970 0.2977 0.2984 0.2991 0.2998 0.3005 0.3012 0.3019 0.3026 0.3033 0.3040 0.3047 0.3054 0.3061 0.3068 0.3075 0.3082 0.3089 0.3096 0.3103 0.3110 0.3117 0.3124 0.3131 0.3138 0.3145 0.3152 0.3159 0.3166 0.3173 0.3180 0.3187 0.3194 0.3201 0.3208 0.3215 0.3222 0.3229 0.3236 0.3243 0.3250 0.3257 0.3264 0.3271 0.3278 0.3285 0.3292 0.3299 0.3306 0.3313 0.3320 0.3327 0.3334 0.3341 0.3348 0.3355 0.3362 0.3369 0.3376 0.3383 0.3390 0.3397 0.3404 0.3411 0.3418 0.3425 0.3432 0.3439 0.3446 0.3453 0.3460 0.3467 0.3474 0.3481 0.3488 0.3495 0.3502 0.3509 0.3516 0.3523 0.3530 0.3537 0.3544 0.3551 0.3558 0.3565 0.3572 0.3579 0.3586 0.3593 0.3600 0.3607 0.3614 0.3621 0.3628 0.3635 0.3642 0.3649 0.3656 0.3663 0.3670 0.3677 0.3684 0.3691 0.3698 0.3705 0.3712 0.3719 0.3726 0.3733 0.3740 0.3747 0.3754 0.3761 0.3768 0.3775 0.3782 0.3789 0.3796 0.3803 0.3810 0.3817 0.3824 0.3831 0.3838 0.3845 0.3852 0.3859 0.3866 0.3873 0.3880 0.3887 0.3894 0.3901 0.3908 0.3915 0.3922 0.3929 0.3936 0.3943 0.3950 0.3957 0.3964 0.3971 0.3978 0.3985 0.3992 0.3999 0.4006 0.4013 0.4020 0.4027 0.4034 0.4041 0.4048 0.4055 0.4062 0.4069 0.4076 0.4083 0.4090 0.4097 0.4104 0.4111 0.4118 0.4125 0.4132 0.4139 0.4146 0.4153 0.4160 0.4167

0 0729=-0.01392

[illegible]

0 00452 0 06622

0.4533% 0.0225% 0.9896% -0.2023% 0.2340% -0.0391% 0.0668% -0.0425%

0.01713 0.07043

0.99 5-0. 4097< 0. 0232< 0. 9270>-0. 1746< 0. 2267>-0. 0745< 0. 1190>-0. 0726<

0.0775> 0.1118>
 499 6 0.3736< 0.0227> 0.8526> 0.1498< 0.2218> 0.1121< 0.1677> 0.1046<
 0.1393> 0.1915>
 799 6 0.1700> 0.0970< 9.9999> 0.1667> 0.4119< 0.0670< 0.3194> 0.2339<
 -0.0561> 0.2165>
 899 6 0.1713> 0.0853< 9.9999> 0.1660> 0.3894< 0.0533< 0.3104> 0.2114<
 -0.0606> 0.1934>
 999 6 0.2083> 0.0822< 9.9999> 0.1854> 0.4277< 0.0581< 0.3334> 0.2406<
 -0.0812> 0.2528>
 1099 6 0.2397> 0.0940< 9.9999> 0.2066> 0.4869< 0.0226> 0.3050> 0.2305<
 -0.1454< 0.3568>
 1199 6 0.2499> 0.0896< 9.9999> 0.2016> 0.4722< 0.0566> 0.0787> 0.1235<
 -0.2300< 0.4888>
 1299 6 0.2121> 0.0976< 9.9999> 0.1740> 0.4324< 0.0377> 0.2087> 0.2150<
 -0.0893< 0.4182>
 1399 6 0.1779> 0.0935< 9.9999> 0.1653> 0.3971< 0.0689< 0.2269> 0.2362<
 -0.0461> 0.3820>
 1499 6 0.1381> 0.0466< 9.9999> 0.1934> 0.2833< 0.0744> 0.0552> 0.0253=&br/>
 -0.2094< 0.2537>
 1599 6 0.1098> 0.0438< 9.9999> 0.1912> 0.2623< 0.0498> 0.0414> 0.0441<
 -0.1825< 0.2262>
 1699 6 0.0723> 0.0573< 9.1913> 0.1731> 0.2379< 0.0089> 0.0087> 0.0771<
 -0.1340< 0.1792>
 1799 6 0.1381> 0.0635< 9.9999> 0.1863> 0.3180< 0.0821> 0.1217> 0.0337<
 -0.2197< 0.4281>
 1899 6 0.0608> 0.0156< 6.9810< 0.1722> 0.1368< 0.0539> 0.1731> 0.0443=&br/>
 -0.1731< 0.0083>
 1999 5 0.1043> 0.1161< 9.9999> 0.0964> 0.3426< 0.1419< 0.2650> 0.2365<
 -0.0300> 0.0575>
 2099 6 0.1050> 0.1028< 9.5115> 0.0938> 0.2904< 0.1922< 0.3360> 0.2631<
 0.0696> 0.1759>
 2199 6 0.1440> 0.1316< 9.9999> 0.1018> 0.4035< 0.0626< 0.1991> 0.1776<
 -0.1515< 0.0232>
 2299 6 0.1384> 0.1180< 9.9999> 0.1007> 0.3456< 0.1401< 0.3050> 0.2291<
 -0.0182> 0.1302>
 2399 6 0.0193> 0.1260< 8.5169> 0.0726> 0.3310< 0.0946< 0.0175> 0.1620<
 -0.0867< 0.1273>
 2499 6 0.1040> 0.1049< 4.8263> 0.0790> 0.2759< 0.1635< 0.1388> 0.2039<
 0.0451> 0.0510>
 2599 6 0.1298> 0.0932> 1.9240> 0.0311> 0.1020> 0.2154< 0.6030> 0.2136<
 0.2509> 0.6084>
 2699 6 0.1189> 0.0111> 3.0883> 0.0223> 0.0756> 0.1767< 0.5645> 0.1863<
 0.1968> 0.5500>
 2799 6 0.0135> 0.0106> 3.8301> 0.0152> 0.0064< 0.0561< 0.4924> 0.0813<
 0.0550> 0.4964>
 2899 6 0.0313> 0.0159> 5.1973> 0.0001> 0.0211< 0.0199> 0.4498> 0.0569<
 -0.0073> 0.4323>
 2999 6 0.0324> 0.0188> 5.8235> 0.0083> 0.0407< 0.0269> 0.4173> 0.0181=&br/>
 -0.0627> 0.4051>
 3099 6 0.0554> 0.0240> 6.4517> 0.0172> 0.0697< 0.0227> 0.3854> 0.0111=&br/>
 -0.1166> 0.3682>
 3199 6 0.0544> 0.0259> 7.3229> 0.0268> 0.0801< 0.0740> 0.3667> 0.0165=&br/>
 -0.1386< 0.5396>
 3299 6 0.0659> 0.0212> 2.9334> 0.0530> 0.0515< 0.0085> 0.1494> 0.0489<
 -0.0690> 0.1588>
 3399 6 0.0604> 0.0259> 2.8408> 0.0519> 0.0527< 0.0019> 0.1403> 0.0387<
 -0.0801> 0.1343>
 3499 6 0.0592> 0.0303> 2.8087> 0.0523> 0.0546< 0.0039> 0.1400> 0.0365<
 -0.0814> 0.1349>
 3599 6 0.0521> 0.0062> 4.2460> 0.0653> 0.0694< 0.0175> 0.1186> 0.0322<

-0.1123< 0.0894<
 369 6-0.0531< 0.0071<
 -0.1176< 0.0826<
 379 6-0.0360< 0.0069<
 -0.1293< 0.0821<
 389 6-0.0272< 0.0071<
 -0.1377< 0.0835<
 399 6-0.0281< 0.0120<
 -0.1480< 0.0307<
 409 6 0.0301< 0.0298<
 -0.2003< 0.0987<
 419 6 0.0369< 0.0322<
 -0.3142< 0.1663<
 429 6 0.0635< 0.0352<
 -0.3431< 0.1696<
 439 6 0.0846< 0.0373<
 -0.3646< 0.1718<
 449 6 0.0574< 0.0423<
 -0.3739< 0.2537<
 459 6 0.0451< 0.0404<
 -0.3565< 0.2467<
 469 6 0.0797< 0.0442<
 -0.3924< 0.2506<
 479 6 0.0349< 0.0675<
 0.2159< 0.3821<
 489 6 0.1979< 0.1144<
 -0.1165< 0.0791<
 499 6 0.0584< 0.0731<
 0.1798< 0.3607<
 509 6 0.1133< 0.0984<
 0.0307< 0.2294<
 519 6 0.1552< 0.1088<
 -0.0554< 0.1428<
 4.3874< 0.0657#-0.0706< 0.0208> 0.1139>-0.0300<
 4.3363> 0.0657#-0.0777< 0.0336> 0.1101>-0.0178#
 4.0656> 0.0646#-0.0795< 0.0433> 0.1046>-0.0074#
 5.6126> 0.0775#-0.0913< 0.0563> 0.0790>-0.0037#
 8.8765> 0.1100#-0.1401< 0.1353>-0.0006> 0.0526#
 9.9999> 0.1254#-0.1545< 0.1532>-0.0316> 0.0597>
 9.9999> 0.1251#-0.1620< 0.1837>-0.0489> 0.0903>
 9.9999> 0.1258#-0.1696< 0.2059>-0.0603> 0.1121>
 9.9999> 0.1406>-0.1728< 0.1967>-0.0856> 0.0916>
 9.9999> 0.1403>-0.1684< 0.1803>-0.0741> 0.0758>
 9.9999> 0.1414>-0.1803< 0.2173>-0.0936> 0.1122>
 5.8717> 0.0725#-0.1510<-0.2463< 0.4915>-0.2959<
 9.9999> 0.2009>-0.5491<-0.1516< 0.4921>-0.3445<
 7.3435> 0.0789#-0.1881<-0.2289< 0.4907>-0.2907<
 9.9999> 0.1340#-0.3716<-0.1962< 0.5008>-0.3257<
 9.9999> 0.1727>-0.4772<-0.1720< 0.4987>-0.3396<

ENTER NAME OF NEXT POPULATION FILE (' ' IF NONE)

51 POINTS ON REVISED SAMPLE FILE
 0 NON REDUNDANT FROM INPUT SAMPLE FILE
 51 FROM POPLATN FILES, WITH ID NOT IN SAMPLE FILE
 0 FROM POPLATN FILES, WITH ID IN SAMPLE FILE
 0 FROM POPLATN FILES, CORRECTLY CLASSIFIED
 0 FROM POPLATN FILE, NOT CORRECTLY CLASSIFIED
 0 CORRECTABLE, UNIQUE, USING GAPS
 PER SET: 0 0 0 0 0
 0 CORRECTABLE, NOT UNIQUE, USING GAPS
 PER SET: 0 0 0 0 0
 0 NO CLASSIFICATION, USING GAPS
 PER SET: 0 0 0 0 0
 0 INCORRECT CLASS, NOT UNIQUE, USING GAPS
 PER SET: 0 0 0 0 0
 0 INCORRECT CLASS, UNIQUE, USING GAPS
 PER SET: 0 0 0 0 0
 0 INCORRECT CLASS, UNIQUE, WITHOUT GAPS
 PER SET: 0 0 0 0 0

POPULATION REVISED SAMPLE FILE (' ' OLD SETS. 1 NEW SETS)
 SET ID 10 7 22 5 23 99
 POINTS PER SET 0 0 0 0 0 51

HYPERPLANES FOR SET DISCRIMINATION, IN TERMS OF INPUT VARIABLES
 HYPERPLANES FOR SETS 1(-0.2332E+00 LE RHS)& 2(-0.2061E+00 GE RHS)
 RHS IS - 0.187E-02*(1) + 0.481E-03*(2) - 0.917E-01*(3)
 + 0.583E-02*(4) - 0.165E-01*(5) + 0.458E+00*(6) - 0.275E-03*(7)
 - 0.725E-01*(8) - 0.211E-01*(9)
 HYPERPLANES FOR SETS 1(0.1029E+00 LE RHS)& 3(0.1109E+00 GE RHS)
 RHS IS + 0.490E-03*(1) + 0.194E-03*(2) - 0.116E-01*(3)
 - 0.341E-02*(4) + 0.106E-02*(5) + 0.645E+00*(6) - 0.541E-03*(7)
 - 0.413E-02*(8) + 0.247E-01*(9)
 HYPERPLANES FOR SETS 1(-0.1976E+01 LE RHS)& 4(-0.1976E+01 GE RHS)
 RHS IS + 0.584E+00*(1) - 0.185E-01*(2) - 0.829E+01*(3)
 + 0.626E+00*(4) + 0.123E+00*(5) - 0.865E+00*(6) - 0.151E-02*(7)
 - 0.601E+00*(8) + 0.235E+01*(9)
 HYPERPLANES FOR SETS 1(0.2736E+00 LE RHS)& 5(0.5548E+00 GE RHS)
 RHS IS - 0.150E-02*(1) + 0.358E-03*(2) + 0.203E+00*(3)
 + 0.321E-02*(4) + 0.133E-02*(5) + 0.62E+00*(6) - 0.445E-03*(7)
 - 0.432E-01*(8) + 0.847E-02*(9)
 HYPERPLANES FOR SETS 2(-0.2196E+00 LE RHS)& 3(-0.2448E+00 GE RHS)
 RHS IS + 0.179E-02*(1) - 0.382E-03*(2) - 0.239E-01*(3)
 - 0.130E-01*(4) + 0.639E-02*(5) - 0.101E+00*(6) - 0.596E-04*(7)
 + 0.121E-01*(8) + 0.987E-01*(9)
 HYPERPLANES FOR SETS 2(0.2993E-01 LE RHS)& 4(0.2959E-01 GE RHS)
 RHS IS + 0.415E-02*(1) - 0.183E-03*(2) - 0.450E-01*(3)
 - 0.535E-02*(4) - 0.126E-01*(5) + 0.292E+00*(6) + 0.150E-03*(7)
 + 0.114E-01*(8) - 0.896E-01*(9)
 HYPERPLANES FOR SETS 2(-0.2412E+01 LE RHS)& 5(-0.2408E+01 GE RHS)
 RHS IS - 0.462E-02*(1) - 0.116E-02*(2) - 0.617E+00*(3)
 + 0.115E-01*(4) + 0.396E-02*(5) - 0.149E+00*(6) - 0.165E-03*(7)
 - 0.436E-01*(8) + 0.552E-01*(9)
 HYPERPLANES FOR SETS 3(0.1233E+00 LE RHS)& 4(0.3780E-01 GE RHS)
 RHS IS + 0.224E-02*(1) - 0.359E-04*(2) - 0.332E-01*(3)
 - 0.946E-02*(4) - 0.124E-01*(5) + 0.253E+00*(6) + 0.138E-03*(7)
 + 0.137E-01*(8) - 0.917E-01*(9)
 HYPERPLANES FOR SETS 3(-0.4660E+00 LE RHS)& 5(-0.4865E+00 GE RHS)
 RHS IS - 0.157E-02*(1) + 0.133E-03*(2) - 0.483E-01*(3)
 + 0.504E-03*(4) + 0.102E-01*(5) - 0.413E+00*(6) - 0.236E-03*(7)
 - 0.159E-01*(8) + 0.839E-01*(9)
 HYPERPLANES FOR SETS 4(-0.2765E+01 LE RHS)& 5(-0.2988E+01 GE RHS)
 RHS IS - 0.179E-01*(1) - 0.103E-02*(2) - 0.564E+00*(3)
 - 0.124E-03*(4) - 0.266E-03*(5) - 0.238E-01*(6) - 0.827E-04*(7)
 - 0.324E-01*(8) - 0.398E-02*(9)

NAME INPUT, OUTPUT HYPERPLANE FILE (, , TO OMIT)

CHUD DAT, 1

CHUD DAT, 1

CHOOSE TO SAVE OLD, NEW PLANES (0,N) FOR SETS 2, 3
 CHOOSING OLD PLANES WITHOUT COMPARISON WITH NEW
 CHOOSE TO SAVE OLD, NEW PLANES (0,N) FOR SETS 2, 4
 CHOOSING OLD PLANES WITHOUT COMPARISON WITH NEW
 CHOOSE TO SAVE OLD, NEW PLANES (0,N) FOR SETS 2, 5
 CHOOSING OLD PLANES WITHOUT COMPARISON WITH NEW
 CHOOSE TO SAVE OLD, NEW PLANES (0,N) FOR SETS 3, 4
 CHOOSING OLD PLANES WITHOUT COMPARISON WITH NEW
 CHOOSE TO SAVE OLD, NEW PLANES (0,N) FOR SETS 3, 5
 CHOOSING OLD PLANES WITHOUT COMPARISON WITH NEW
 CHOOSE TO SAVE OLD, NEW PLANES (0,N) FOR SETS 4, 5
 CHOOSING OLD PLANES WITHOUT COMPARISON WITH NEW
 HYPERPLANE OUTPUT FILE WRITTEN

11.8 USED 39202.24 SECONDS, COMPLETED AT 02 37 HOURS

ENTER 0,N NOTTO,TO TRANSFORM DATA PER CLUSTER N AXES, NCO PER ALL DATA
0
ENTER 0,1,2 AS THERE ARENT,ARE NONE CASES,CHANGE CASE
0

FORTRAN STOP
1 EXIT
CASEY job terminated at 11-APR-1987 22:52:05.60

Accounting information:
Buffered I/O count: 903 Peak working set size: 1068
Direct I/O count: 1637 Peak page file size: 4865
Page faults: 4217451 Mounted volumes: 0
Charged CPU time: 0 10:11:12.16 Elapsed time: 0 10:53:28.31

10	7	-0.2332405	-0.2061299	0.0000000E+00
-1.8727807E-03	4.8079339E-04	0.0000000E+00	0.0000000E+00	-9.1713540E-02
0.0000000E+00	0.0000000E+00	5.8340430E-03	-1.4465910E-02	0.4578562
0.0000000E+00	0.0000000E+00	-2.7470119E-04	-7.2498590E-02	-2.1147290E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
10	22	0.1029203	0.1109052	0.0000000E+00
4.8990099E-04	1.9362812E-04	0.0000000E+00	0.0000000E+00	-1.1586168E-02
0.0000000E+00	0.0000000E+00	-3.4052713E-03	1.0563395E-03	0.4447840
0.0000000E+00	0.0000000E+00	-5.4064224E-04	-4.1272528E-03	2.4662258E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
10	5	-1.975560	-1.975577	0.0000000E+00
0.5842420	-1.8510444E-02	0.0000000E+00	0.0000000E+00	-8.293071
0.0000000E+00	0.0000000E+00	0.6258693	0.1231744	-0.8652425
0.0000000E+00	0.0000000E+00	-1.5126106E-03	-0.6009062	2.351106
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
10	23	0.2736165	0.5548023	0.0000000E+00
-1.4984467E-03	3.5825159E-04	0.0000000E+00	0.0000000E+00	0.2030284
0.0000000E+00	0.0000000E+00	3.2119798E-03	1.3291849E-03	0.6619402
0.0000000E+00	0.0000000E+00	-4.4525723E-04	-4.3248504E-02	8.4720310E-03
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
7	22	0.5795203	0.5227754	0.0000000E+00
3.1011046E-03	-5.0478679E-04	0.0000000E+00	0.0000000E+00	0.3108462
0.0000000E+00	0.0000000E+00	-2.3448415E-02	3.0095249E-02	0.2768602
0.0000000E+00	0.0000000E+00	9.8913180E-05	1.6721459E-02	5.9865196E-03
6.2796712E-02	7.4157298E-02	-5.0872818E-02	-2.3260647E-03	0.0000000E+00
7	5	-1.049361	-1.061939	0.0000000E+00
6.3395808E-03	-8.1227126E-04	0.0000000E+00	0.0000000E+00	-0.1962710
0.0000000E+00	0.0000000E+00	-4.7657855E-02	-1.0935842E-02	1.6853672E-02
0.0000000E+00	0.0000000E+00	-1.3219585E-06	9.0055102E-03	1.2956944E-03
0.2257215	0.9999677	-5.3811789E-02	0.4225681	0.0000000E+00
7	23	-2.303845	-2.518939	0.0000000E+00
-2.7961617E-03	-1.4191014E-03	0.0000000E+00	0.0000000E+00	-0.9289371
0.0000000E+00	0.0000000E+00	1.4517985E-02	6.9554118E-03	0.1378282
0.0000000E+00	0.0000000E+00	3.4872988E-03	1.8711749E-02	6.7446232E-03
-9.9231064E-02	5.4469764E-02	-5.5005670E-02	-3.5575449E-02	0.0000000E+00
22	5	3.9294332E-02	3.8033158E-02	0.0000000E+00
2.0769412E-02	-9.6607284E-04	0.0000000E+00	0.0000000E+00	-4.6494271E-04
0.0000000E+00	0.0000000E+00	1.7119616E-02	-1.7073029E-03	-2.7587023E-02
0.0000000E+00	0.0000000E+00	-8.3054110E-03	-6.6154227E-03	-2.1943608E-03
-8.3126575E-02	1.9225936E-03	-1.3546035E-02	-6.8973869E-02	0.0000000E+00
22	23	-1.673259	-1.693602	0.0000000E+00
-3.7568253E-03	4.1548777E-04	0.0000000E+00	0.0000000E+00	-0.7813047
0.0000000E+00	0.0000000E+00	1.3587160E-02	1.3430241E-02	-0.4653442
0.0000000E+00	0.0000000E+00	-2.6481459E-04	-2.3191137E-02	-5.4212278E-03
6.7232994E-03	-4.3115616E-02	1.8154845E-02	-7.1016550E-02	0.0000000E+00
5	23	-2.250376	-2.682809	0.0000000E+00
-1.0326902E-02	-1.0526507E-03	0.0000000E+00	0.0000000E+00	-0.9193538
0.0000000E+00	0.0000000E+00	8.1318943E-03	7.5071254E-03	0.1464850
0.0000000E+00	0.0000000E+00	6.4787317E-05	2.0933010E-02	7.4759726E-03
0.9078475	0.1877992	5.6158304E-03	-4.6592832E-02	0.0000000E+00
0	7	0.8240990	0.9276105	0.0000000E+00
-1.1082066E-03	1.8902248E-04	0.0000000E+00	0.0000000E+00	0.2249505
0.0000000E+00	0.0000000E+00	2.0155622E-02	1.9414650E-02	0.2750238
0.0000000E+00	0.0000000E+00	4.4288008E-05	-4.0131032E-02	6.5958038E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
0	22	21.73051	21.73053	0.0000000E+00
0.7826716	-4.3509316E-02	0.0000000E+00	0.0000000E+00	23.32464
0.0000000E+00	0.0000000E+00	0.8472973	0.1072113	-0.1456653
0.0000000E+00	0.0000000E+00	-3.644865E-03	0.1075505	-3.3292517E-02
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00

0	5	28.22051	28.22050	0.000000E+00
1.021026	-5.686132E-02	0.000000E+00	0.000000E+00	30.42640
0.000000E+00	0.000000E+00	1.114380	0.1269589	-0.3560690
0.000000E+00	0.000000E+00	-4.640508E-03	0.1440398	-4.0388469E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
0	23	0.2599925	0.2599437	0.000000E+00
-2.420747E-05	2.796578E-04	0.000000E+00	0.000000E+00	3.426772E-02
0.000000E+00	0.000000E+00	2.579820E-04	3.324808E-03	0.5580868
0.000000E+00	0.000000E+00	-4.897021E-04	-1.163508E-02	5.171754E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
2	3	-0.2197392	-0.2180968	0.000000E+00
7.735032E-04	-2.715618E-04	0.000000E+00	0.000000E+00	-1.150962E-02
0.000000E+00	0.000000E+00	-3.683989E-03	-1.413649E-03	-0.4094222
0.000000E+00	0.000000E+00	3.980340E-04	1.503869E-02	-7.685698E-02
-0.2434935	-0.1101012	-4.154473E-03	-1.713275E-03	0.000000E+00
2	20	-0.5007082	-0.5007089	0.000000E+00
2.363279E-03	-6.218868E-04	0.000000E+00	0.000000E+00	-5.119301E-02
0.000000E+00	0.000000E+00	-1.153518E-02	1.395278E-02	-0.5413191
0.000000E+00	0.000000E+00	4.696815E-04	-1.725956E-02	2.591567E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
2	4	-0.9735251	-0.9924156	0.000000E+00
-2.011424E-05	8.208629E-05	0.000000E+00	0.000000E+00	-1.170877
0.000000E+00	0.000000E+00	3.738597E-03	8.056141E-03	0.1476528
0.000000E+00	0.000000E+00	1.662001E-04	1.931074E-02	8.244636E-03
2.506419E-02	-9.799057E-02	-1.896509E-02	5.906591E-02	0.000000E+00
3	20	-5.048846E-02	-5.048771E-02	0.000000E+00
-5.718573E-04	-1.439465E-04	0.000000E+00	0.000000E+00	2.765118E-03
0.000000E+00	0.000000E+00	4.521194E-03	5.223870E-03	-0.3373621
0.000000E+00	0.000000E+00	2.606674E-04	3.426772E-02	-8.578571E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
3	4	-1.060428	-1.079199	0.000000E+00
8.235081E-04	5.191356E-05	0.000000E+00	0.000000E+00	-1.190682
0.000000E+00	0.000000E+00	6.122337E-03	8.955590E-03	8.819414E-02
0.000000E+00	0.000000E+00	1.229394E-04	1.539317E-02	7.081095E-03
-3.163728E-02	4.129334E-02	-0.2750822	1.000090	0.000000E+00
20	4	1.297541E-02	1.297463E-02	0.000000E+00
-3.279981E-04	-7.679725E-05	0.000000E+00	0.000000E+00	-4.369946E-03
0.000000E+00	0.000000E+00	4.896197E-05	-4.701188E-03	-4.880049E-02
0.000000E+00	0.000000E+00	1.662263E-04	-3.924547E-02	9.660907E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
1	2	0.5111591	0.5483046	0.000000E+00
-6.838637E-04	4.019597E-04	0.000000E+00	0.000000E+00	3.959065E-02
0.000000E+00	0.000000E+00	7.277214E-03	7.315898E-03	0.5746222
0.000000E+00	0.000000E+00	-4.267763E-04	2.551409E-02	-4.921614E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
1	3	0.2963204	0.2965649	0.000000E+00
-6.431413E-04	3.634970E-04	0.000000E+00	0.000000E+00	-6.677198E-04
0.000000E+00	0.000000E+00	4.495659E-03	1.489846E-03	0.4907922
0.000000E+00	0.000000E+00	-3.796382E-04	2.202252E-02	-7.014410E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
1	4	0.3013018	0.3013037	0.000000E+00
-7.793255E-04	3.810644E-04	0.000000E+00	0.000000E+00	-1.710324E-03
0.000000E+00	0.000000E+00	6.057922E-03	6.002986E-04	0.6357226
0.000000E+00	0.000000E+00	-5.478794E-04	1.572603E-03	1.873127E-02
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00
2	10	-0.1649746	-0.1634048	0.000000E+00
-5.523511E-05	-3.178681E-04	0.000000E+00	0.000000E+00	-1.065527E-02
0.000000E+00	0.000000E+00	5.866959E-04	1.709523E-03	-0.6418737
0.000000E+00	0.000000E+00	5.700612E-04	1.092226E-03	5.672590E-04
0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00	0.000000E+00

3 10 0. 2472814 0. 2473280 0. 0000000E+00
-1. 565633E-03 -1. 720847E-04 0. 0000000E+00 0. 0000000E+00 6. 7100063E-02
0. 0000000E+00 0. 0000000E+00 9. 6575325E-03 1. 4601735E-02 -0. 5838240
0. 0000000E+00 0. 0000000E+00 4. 9647613E-04 2. 4369385E-02 4. 4312496E-03
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
10 4 0. 3886607 0. 4994883 0. 0000000E+00
-1. 1849942E-03 2. 9601471E-04 0. 0000000E+00 0. 0000000E+00 0. 3216445
0. 0000000E+00 0. 0000000E+00 1. 4153545E-03 1. 4043109E-03 0. 6345115
0. 0000000E+00 0. 0000000E+00 -2. 8472478E-04 -6. 8457232E-02 8. 4627448E-03
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
0 2 0. 1542452 0. 2216018 0. 0000000E+00
4. 5465229E-03 -2. 9410623E-04 0. 0000000E+00 0. 0000000E+00 4. 6789087E-02
0. 0000000E+00 0. 0000000E+00 -2. 1083990E-02 1. 2364086E-02 0. 4006078
0. 0000000E+00 0. 0000000E+00 -1. 1092038E-04 3. 1567797E-02 -7. 0265412E-02
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
0 3 0. 1211064 0. 2582814 0. 0000000E+00
2. 4283908E-03 -1. 5722604E-04 0. 0000000E+00 0. 0000000E+00 1. 3884456E-02
0. 0000000E+00 0. 0000000E+00 -1. 2961977E-02 1. 4028946E-02 0. 3762399
0. 0000000E+00 0. 0000000E+00 -1. 2358476E-04 3. 2897972E-02 -7. 8430772E-02
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
0 4 0. 1950551 0. 1994090 0. 0000000E+00
-4. 8026396E-04 3. 1156171E-04 0. 0000000E+00 0. 0000000E+00 -5. 1489808E-03
0. 0000000E+00 0. 0000000E+00 3. 1105785E-03 1. 0997916E-04 0. 6633888
0. 0000000E+00 0. 0000000E+00 -5. 4658845E-04 -4. 5761033E-03 9. 8880753E-03
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
21 1 -0. 1713047 -0. 1713012 0. 0000000E+00
-2. 0414009E-03 -1. 8875749E-04 0. 0000000E+00 0. 0000000E+00 5. 1234782E-02
0. 0000000E+00 0. 0000000E+00 7. 7920733E-03 -8. 8252081E-03 -0. 5322345
0. 0000000E+00 0. 0000000E+00 3. 0975265E-04 -7. 9713175E-03 7. 0511021E-02
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
21 20 -7. 2240822E-02 -4. 2462621E-02 0. 0000000E+00
-1. 9702390E-03 -1. 1756549E-04 0. 0000000E+00 0. 0000000E+00 6. 5879524E-02
0. 0000000E+00 0. 0000000E+00 1. 11329865E-02 -3. 2998407E-03 -0. 7314168
0. 0000000E+00 0. 0000000E+00 4. 5278388E-04 1. 4669186E-02 -1. 3945083E-04
-8. 4402657E-03 -6. 8077490E-02 -9. 0668641E-02 -6. 3659482E-02
1 20 0. 2794988 0. 2794958 0. 0000000E+00
4. 5013024E-03 4. 1882688E-03 0. 0000000E+00 0. 0000000E+00 -1. 2021876E-02
0. 0000000E+00 0. 0000000E+00 -1. 0945338E-02 1. 0942113E-02 0. 6504494
0. 0000000E+00 0. 0000000E+00 -4. 2033434E-04 1. 1498610E-02 -3. 1931791E-02
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
21 10 -0. 4329941 -0. 2422430 0. 0000000E+00
2. 3396760E-03 -3. 8041692E-04 0. 0000000E+00 0. 0000000E+00 -0. 1273370
0. 0000000E+00 0. 0000000E+00 -7. 3157344E-03 -1. 4265542E-03 -0. 6231833
0. 0000000E+00 0. 0000000E+00 4. 8975350E-04 2. 7889628E-02 -2. 7262237E-02
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
20 10 -0. 4110589 -0. 1805754 0. 0000000E+00
3. 4529907E-03 -3. 0843678E-04 0. 0000000E+00 0. 0000000E+00 -0. 2007786
0. 0000000E+00 0. 0000000E+00 -8. 2077283E-03 -2. 1495740E-03 -0. 6030180
0. 0000000E+00 0. 0000000E+00 4. 5786559E-04 4. 0426899E-02 -2. 5238711E-02
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
0 20 23. 07658 24. 38633 0. 0000000E+00
1 235638 -5. 5915526E-02 0. 0000000E+00 0. 0000000E+00 42. 03928
0. 0000000E+00 0. 0000000E+00 0. 5945719 0. 7504909
0. 0000000E+00 0. 0000000E+00 -1. 3030957E-02 -0. 6719638 -0. 2679172
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00
21 0 -20. 16562 -20. 16585 0. 0000000E+00
-1. 397251 6. 9023222E-02 0. 0000000E+00 0. 0000000E+00 -42. 86494
0. 0000000E+00 0. 0000000E+00 -0. 7038303 -0. 9178057 12. 28921
0. 0000000E+00 0. 0000000E+00 1. 3084889E-02 1. 110618 0. 3762960
0. 0000000E+00 0. 0000000E+00 0. 0000000E+00 0. 0000000E+00

1	10	0.4961410	0.5280035	0.0000000E+00	0.0000000E+00
-1.8672388E-04	5.2940213E-05	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.1234468
0.0000000E+00	0.0000000E+00	7.4593048E-03	1.1848936E-02	-1.3010859E-02	
0.0000000E+00	0.0000000E+00	1.2015631E-04	4.2882677E-02	-9.0552568E-02	
0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00	
1	0	0.2336065	0.2568044	0.0000000E+00	
-1.4813519E-03	3.835839E-04	0.0000000E+00	0.0000000E+00	-1.1672183E-02	
0.0000000E+00	0.0000000E+00	7.0010051E-03	7.3218596E-04	0.3321991	
0.0000000E+00	0.0000000E+00	-2.7845841E-04	2.6801050E-02	-8.6197518E-02	
-2.2288227E-02	-2.2170909E-02	-9.0032112E-04	-1.8296598E-02		
0	10	-0.2321022	2.5051221E-02	0.0000000E+00	
3.9048633E-03	-2.9241768E-04	0.0000000E+00	0.0000000E+00	-8.2190022E-02	
0.0000000E+00	0.0000000E+00	-8.3781667E-03	4.4831430E-04	-0.4609540	
0.0000000E+00	0.0000000E+00	2.9259894E-04	6.3622579E-02	-5.5368930E-02	
-7.4079605E-03	1.8480514E-03	2.6887782E-02	3.0446211E-02		
2	7	7.3668428E-02	7.3668830E-02	0.0000000E+00	
-1.7890703E-03	3.3631400E-04	0.0000000E+00	0.0000000E+00	-6.2551722E-02	
0.0000000E+00	0.0000000E+00	1.0815877E-02	-5.4139541E-03	0.1017401	
0.0000000E+00	0.0000000E+00	-8.5408981E-05	1.9421577E-02	-9.8950692E-02	
-5.5741403E-02	-5.6583881E-03	9.2840791E-03	-3.6627650E-03		
2	22	0.3944690	0.4119532	0.0000000E+00	
1.5834599E-03	1.0278248E-04	0.0000000E+00	0.0000000E+00	1.7171111E-02	
0.0000000E+00	0.0000000E+00	1.8757958E-02	6.3992417E-03	0.3582918	
0.0000000E+00	0.0000000E+00	-3.4819610E-04	-5.3887633E-03	7.6162145E-02	
0.2500579	-8.4303617E-03	-4.4833183E-02	-7.9101473E-02		
2	5	0.8189578	0.7408233	0.0000000E+00	
5.4531274E-03	-2.3480231E-04	0.0000000E+00	0.0000000E+00	-0.1359558	
0.0000000E+00	0.0000000E+00	-1.1981551E-02	-7.8201024E-03	0.6144885	
0.0000000E+00	0.0000000E+00	4.2819351E-04	3.9935797E-02	1.2465645E-02	
1.4725924E-03	-0.2443150	-4.7987498E-02	-1.4814559E-02		
2	23	-1.998477	-2.335875	0.0000000E+00	
-2.0845931E-03	-1.2637860E-03	0.0000000E+00	0.0000000E+00	-0.9638517	
0.0000000E+00	0.0000000E+00	1.1659215E-02	6.1305990E-03	0.2097973	
0.0000000E+00	0.0000000E+00	9.9564073E-05	2.3359001E-02	8.3339429E-03	
-7.9829693E-03	0.1029508	-5.7595670E-02	0.3869254		
3	7	0.1586641	0.1524081	0.0000000E+00	
-3.6818380E-03	9.5083250E-04	0.0000000E+00	0.0000000E+00	-0.1904521	
0.0000000E+00	0.0000000E+00	3.2595232E-02	-2.4979820E-02	-0.2175574	
0.0000000E+00	0.0000000E+00	-4.4529233E-05	-1.6139196E-02	-5.1972037E-03	
1.4568806E-02	-1.2550235E-02	3.9228767E-02	8.7366700E-02		
3	22	-0.1314286	-0.1528783	0.0000000E+00	
7.7515920E-03	-9.7303135E-04	0.0000000E+00	0.0000000E+00	5.5731092E-02	
0.0000000E+00	0.0000000E+00	-3.5994649E-02	1.8355209E-02	0.2615914	
0.0000000E+00	0.0000000E+00	9.3401897E-05	1.9759661E-02	6.1906716E-03	
-7.5340152E-02	-0.1518413	1.6444057E-02	-2.3264112E-03		
3	5	-2.148834	-2.170104	0.0000000E+00	
-1.3511101E-04	-1.5010128E-03	0.0000000E+00	0.0000000E+00	-0.9038910	
0.0000000E+00	0.0000000E+00	1.6256787E-02	6.5549118E-03	0.1306692	
0.0000000E+00	0.0000000E+00	2.3591552E-05	1.7382551E-02	6.2891878E-03	
-7.6024982E-02	-1.1180199E-02	-6.0328543E-02	-5.8704596E-02		
3	23	-1.859904	-2.158417	0.0000000E+00	
1.4576917E-03	-1.3715932E-03	0.0000000E+00	0.0000000E+00	-0.9311829	
0.0000000E+00	0.0000000E+00	1.3946485E-02	5.5924566E-03	0.2010224	
0.0000000E+00	0.0000000E+00	8.5209823E-05	2.1653350E-02	7.7461298E-03	
-9.5469513E-02	0.2336585	-6.5951347E-02	2.0860612E-02		
7	4	0.6376910	0.6375058	0.0000000E+00	
1.3573484E-04	1.8903003E-04	0.0000000E+00	0.0000000E+00	0.3010593	
0.0000000E+00	0.0000000E+00	-2.0278781E-04	3.8731347E-03	0.4360211	
0.0000000E+00	0.0000000E+00	2.2137816E-04	-9.4285466E-02	-8.8974442E-03	
2.5655508E-02	1.9224125E-03	4.4610351E-03	1.1739350E-02		

22	4	0.6798738	0.6720510	0.0000000E+00
-4.261723E-03	5.7044980E-04	0.0000000E+00	0.0000000E+00	0.1892032
0.0000000E+00	0.0000000E+00	3.0309694E-02	-1.4207790E-02	0.2523862
0.0000000E+00	0.0000000E+00	1.6575288E-04	-7.0536904E-02	-8.1682298E-03
-0.1090021	-9.9959731E-02	-3.3849757E-02	-0.4187044	
4	5	1.861675	1.858364	0.0000000E+00
1.9169506E-02	1.4478797E-03	0.0000000E+00	0.0000000E+00	-0.1949757
0.0000000E+00	0.0000000E+00	1.6165713E-03	2.2439579E-04	-9.4212934E-02
0.0000000E+00	0.0000000E+00	5.7865065E-05	-1.0996808E-02	-1.9000837E-03
-8.9843273E-02	-0.1627179	-6.6109002E-02	-4.3782771E-02	
4	23	-2.263177	-2.450073	0.0000000E+00
-1.3261826E-03	-1.4671807E-03	0.0000000E+00	0.0000000E+00	-0.9167992
0.0000000E+00	0.0000000E+00	1.5509252E-02	6.7464635E-03	0.1341186
0.0000000E+00	0.0000000E+00	2.8684351E-05	1.8010058E-02	6.5048225E-03
-9.1089845E-02	-5.9430838E-02	-0.2651941	-3.0454606E-02	

Appendix F - Sample EFAGHY Input

VAX4> TYPE EFI1.DAT;34
1 '55777B' 20 'Y'
'FLPLANES'
, ,
0 0
0 0
0 0
00111
0
VAX4> LO

Appendix G - Sample EFAGHY Output

First is presented the output that resulted from the batch input of Appendix F.

Then the output from a time sharing session of a slightly later version of EFAGHY. Being in time sharing, the input is an integral part of the output.

```

VAX4>
VAX4>
VAX4> TYPE EFO1.LOG;1
$ !      SYSLOGIN.COM  system-wide standard user login command file.
$ !
$ !      This version is from the DEO VAX cluster in building 200 at column
$ !      This file will need to be customized for other systems in most
$ !      cases.  Any questions may be directed to Ric West at ext. 6-2630.
$ ! Set up the logicals and symbols for Patran2
$ @PATRAN$DIR:PATNAMES
$ VERIFY=1
$ SET NOVERIFY
$ EXIT
$ !
$START:
$ ON CONTROL+Y THEN GOTO START  ! cannot control Y out of this procedure
$ SET NOON
$ NODE = F$GETSYI("NODENAME")
$ SAVE+VERIFY = F$VERIFY(0)
$ !
$ !      set up user enviroment
$ !
$ PROMPT = F$LOGICAL("NODE+NAM")+> "
$ SET PROMPT = DEO1>
$ SET CONTROL=(T,Y)
$ !
$EXIT:
$ @LOGIN
$ !      LOGIN.COM
$ !
$ !      User default login command file for VAX DEO1.  Inserted by
$ !      NEWU.COM when a new user is added to the system.
$ !      Add user features below this line:
$ !
$ EXIT
$ ASSIGN EF11.DAT FOR005
XDCL-I-SUPERSEDE, previous value of FOR005 has been superseded
$ RUN EFAGHY
=====

```

EFAGHY (EMPIRICAL FLUTTER ANALYSIS BY GALACTIC HYPERPLANES)
 RUN ON 05/22/87 AT 16.75 HOURS ON BATCH

ENTER NUMBER & NAMES OF DATA FILES,WORDS/RECORD, Y/N TO CLEARFILES
 1 55777B
 20 Y

NAME GALACTIC HYPERPLANE FILE
 FLPLANES

NAME OUTPUT FILE FOR SET ASSIGNMENTS , ' ' TO PRINT

ENTER N (LE 14) THEN ID OF N SETS TO BE EXCLUDED

0
 ENTER N, THEN N PAIRS OF IDS TO BE LABELED AS THE FIRST
 0

ENTER N THEN THE N HYPERPLANES TO BE USED(0 0 FOR ALL)
 ENTER 0 OR 1 TO NOTWRITE OR WRITE THE FOLLOWING OUTPUT OPTIONS
 HYP,PTS,VOT,ASS,ASP

PT# ACTID CANID YVOTE NVOTE OVOTE GVOTE CVOTE MAX%O MAX%G WVOTE BVOTE

DATA FROM FILE 55777B

1	21	21	10*	2	1	0	9	0.7	0.0	10.7*	8.3*
		0	7	4	2	0	5	1.0	0.0	8.8	4.5
		1	4	6	2	1	1	0.7	0.8	4.9	-3.1
		2	8	4	0	1	5	0.0	0.2	8.8	4.6

3	5	7	0	1	-1	0.0	0.1	5.9	-1.2
4	6	4	1	2	5	0.1	0.7	7.3	1.7
5	1	11	0	1	-9	0.0	0.6	1.4	-10.1
6	1	8	0	4	-3	0.0	0.7	2.9	-7.2
7	3	7	1	2	-1	0.3	0.6	4.7	-3.6
10	7	0	6	0	13*	0.9	0.0	9.3	5.7
20	5	4	1	3	5	0.8	0.8	7.3	1.6
22	9	3	1	0	7	0.6	0.0	9.6	6.2
23	1	4	1	7	5	0.7	1.0	4.6	-3.7
27	3	6	0	4	1	0.0	0.8	4.2	-3.5
2	2	11*	0	2	0	0.3	0.0	11.3	9.7
1	1	10	0	3	0	1.0	0.0	12.1*	11.2*
0	9	0	4	0	13*	0.8	0.0	11.8	10.4
3	10	2	1	0	9	0.4	0.0	10.4	7.9
4	3	9	0	1	-5	0.0	0.8	3.2	-6.7
5	1	10	0	2	-7	0.0	0.8	1.9	-9.1
6	3	7	0	3	-1	0.0	0.9	4.2	-4.5
7	5	6	2	0	1	0.5	0.0	5.6	-1.8
10	4	4	5	0	5	0.5	0.0	5.8	-1.3
20	6	5	2	0	3	0.9	0.0	7.4	2.2
21	2	6	2	3	1	0.6	0.7	4.5	-4.0
22	5	8	0	0	-3	0.0	0.0	5.0	-3.0
23	0	10	1	2	-7	0.6	0.8	1.1	-10.7
27	4	6	0	3	1	0.0	0.3	6.4	-0.2
3	0	10*	0	3	13*	0.8	0.0	12.3*	11.6*
0	1	10*	2	1	9	0.9	0.0	10.9	8.9
2	8	2	2	1	9	0.9	0.7	9.3	5.5
3	9	2	2	0	9	0.2	0.0	9.4	5.7
4	4	7	1	1	-1	0.2	0.0	5.2	-2.6
5	1	10	0	2	-7	0.0	0.4	2.3	-8.3
6	3	5	0	5	3	0.0	0.9	5.8	-1.4
7	9	3	0	1	7	0.0	0.5	9.5	6.0
10	4	4	5	0	5	0.7	0.0	5.1	-0.3
20	2	7	1	3	-1	0.7	0.6	4.8	-3.5
21	1	9	1	2	-5	0.6	0.7	2.1	-8.7
22	6	6	1	0	1	0.3	0.0	6.3	-0.4
23	0	9	1	3	-5	0.6	0.8	1.8	-9.4
27	4	5	0	4	3	0.0	1.0	5.2	-2.6
4	3	11*	1	1	11	0.4	0.0	11.4	9.7
2	2	9	2	0	9	0.2	0.0	9.2	5.3
0	9	0	4	0	13*	1.0	0.0	12.2*	11.3*
1	10	1	2	0	11	1.0	0.0	11.0	9.1
4	4	8	0	1	-3	0.0	0.5	4.5	-4.0
5	1	10	0	2	-7	0.0	0.5	2.1	-8.9
6	3	7	0	3	-1	0.0	0.7	4.7	-3.6
7	7	4	1	1	5	0.9	0.9	8.1	3.2
10	4	3	6	0	7	0.4	0.0	5.5	-2.0
20	3	7	2	1	-1	1.0	0.0	5.6	-1.7
21	1	8	2	2	-3	0.6	0.6	2.6	-7.9
22	6	6	1	0	1	0.9	0.0	6.9	0.8
23	0	10	1	2	-7	0.6	0.8	1.1	-10.7
27	4	5	0	4	3	0.0	1.0	6.1	-0.9
5	0	10*	0	3	13*	1.0	0.0	12.5*	11.9*
1	9	2	1	1	9	0.9	0.6	10.4	7.7
2	7	3	2	1	7	0.3	0.0	8.4	3.7
3	9	2	2	0	9	0.7	0.0	9.9	6.8
4	6	6	1	0	1	0.0	0.0	6.0	-0.9
5	1	10	0	2	-7	0.0	0.4	2.4	-8.1
6	4	4	0	5	5	0.0	0.7	6.9	0.8
7	9	3	0	1	7	0.0	0.3	9.7	6.4
10	5	4	4	0	5	0.4	0.0	6.5	0.0
20	2	8	1	2	-3	0.7	0.3	4.1	-4.8
21	1	9	1	2	-5	0.6	0.9	1.8	-9.3
22	4	8	0	1	-3	0.0	0.3	4.7	-3.6
23	1	8	1	3	-3	0.6	1.0	2.2	-8.6
27	4	5	0	4	3	0.0	0.8	5.5	-2.0

6	21	21	10*	2	1	0	9	0.8	0.0	10.8	8.5
		0	9	0	4	0	13*	1.0	0.0	11.2*	9.5*
		1	6	5	1	1	3	0.4	0.4	7.1	1.1
		2	6	5	2	0	3	0.9	0.0	7.4	1.9
		3	9	2	1	1	9	0.9	0.7	10.1	7.2
		4	3	9	1	0	-5	0.8	0.0	3.8	-5.5
		5	1	11	0	1	-9	0.0	0.3	1.7	-9.6
		6	3	9	0	1	-5	0.0	0.3	3.7	-5.6
		7	5	7	1	0	-1	0.1	0.0	5.1	-2.9
		10	2	5	6	0	3	0.4	0.0	3.1	-6.9
		20	8	3	1	1	7	1.0	0.6	9.4	5.7
		22	5	6	1	1	1	0.6	0.3	6.3	-0.4
		23	0	10	1	2	-7	0.9	0.7	1.4	-10.1
		27	9	2	0	2	9	0.0	0.6	10.0	7.0
7	1	1	11*	0	2	0	13*	0.1	0.0	11.1	9.2
		0	9	0	4	0	13*	0.9	0.0	12.3*	11.6*
		2	9	2	1	1	9	0.1	0.8	9.3	5.6
		3	9	3	1	0	7	0.4	0.0	9.4	5.7
		4	3	10	0	0	-7	0.0	0.0	3.0	-7.0
		5	1	10	0	2	-7	0.0	1.0	1.8	-9.5
		6	3	9	0	1	-5	0.0	0.2	3.8	-5.6
		7	6	6	1	0	1	1.0	0.0	7.0	1.0
		10	4	4	5	0	5	0.4	0.0	5.6	-1.9
		20	8	4	1	0	5	0.8	0.0	8.8	4.5
		21	3	8	1	1	-3	0.4	0.0	4.6	-3.8
		22	4	8	1	0	-3	0.4	0.0	4.6	-3.8
		23	0	10	1	2	-7	0.7	0.8	1.1	-10.8
		27	8	4	0	1	5	0.0	0.2	8.8	4.5
8	21	21	10*	1	2	0	11	0.7	0.0	11.1*	9.2*
		0	8	1	4	0	11	0.9	0.0	10.3	7.7
		1	6	6	1	0	1	0.3	0.0	6.3	-0.5
		2	8	3	1	1	7	0.2	0.1	9.0	5.1
		3	7	4	1	1	5	1.0	0.3	8.7	4.5
		4	3	7	2	1	-1	0.5	0.8	3.9	-5.3
		5	1	11	0	1	-9	0.0	0.4	1.6	-9.8
		6	2	8	0	3	-3	0.0	0.9	3.5	-6.0
		7	3	8	2	0	-3	0.5	0.0	3.9	-5.2
		10	4	0	9	0	13*	0.8	0.0	7.3	1.6
		20	8	2	1	2	9	0.9	0.8	9.6	6.2
		22	6	5	2	0	3	0.9	0.0	7.5	2.1
		23	0	10	1	2	-7	0.8	0.7	1.5	-9.9
		27	4	4	0	5	5	0.0	0.9	6.7	0.3
9	0	0	11*	0	2	0	13*	0.9	0.0	12.8*	12.6*
		1	10	2	0	1	9	0.0	0.7	10.3	7.7
		2	4	7	0	2	-1	0.0	0.7	5.2	-2.7
		3	8	2	1	2	9	0.1	0.5	9.5	6.0
		4	7	4	1	1	5	0.1	0.3	7.8	2.6
		5	3	9	0	1	-5	0.0	0.4	3.6	-5.9
		6	5	4	0	4	5	0.0	0.6	7.6	2.1
		7	10	3	0	0	7	0.0	0.0	10.0	7.0
		10	5	3	5	0	7	0.9	0.0	7.1	1.2
		20	2	9	1	1	-5	0.7	0.7	3.0	-7.1
		21	1	10	1	1	-7	0.6	0.4	2.2	-8.6
		22	4	9	0	0	-5	0.0	0.0	4.0	-5.0
		23	1	7	1	4	-1	0.6	1.0	2.4	-8.2
		27	4	6	0	3	1	0.0	0.8	5.6	-1.8
10	0	3	10*	2	1	0	9	0.4	0.0	10.4	7.8
	0	0	8	0	5	0	13*	0.9	0.0	11.4*	9.9*
		1	6	5	2	0	3	0.6	0.0	6.8	0.6
		2	8	2	3	0	9	0.9	0.0	9.7	6.4
		4	4	7	1	1	-1	0.1	0.4	4.7	-3.6
		5	6	6	0	1	1	0.0	0.2	6.8	0.5
		6	4	6	0	3	1	0.0	0.9	5.7	-1.6
		7	9	3	1	0	7	0.2	0.0	9.2	5.4
		10	4	3	6	0	7	0.8	0.0	7.2	1.5
		20	5	6	1	1	1	0.5	0.1	6.4	-0.2

		21	3	8	1	1	-3	0.5	0.8	3.7	-5.6
		22	6	5	2	0	3	0.5	0.0	6.6	0.2
		23	1	10	1	1	-7	0.5	1.0	1.6	-9.9
11	0	27	0	11	0	2	-9	0.0	0.6	0.8	-11.4
	0	0	9*	0	4	0	13*	0.8	0.0	11.4*	10.1*
		3	9*	2	1	1	9	0.4	0.9	9.5	5.9
		1	6	5	2	0	3	0.4	0.0	6.3	0.6
		2	8	2	3	0	9	0.9	0.0	9.7	6.4
		4	2	10	0	1	-7	0.0	0.8	2.2	-9.6
		5	7	5	0	1	3	0.0	0.5	7.5	2.1
		6	3	6	0	4	1	0.0	0.9	5.4	-2.2
		7	8	3	1	1	7	0.2	0.1	9.1	5.2
		10	4	3	6	0	7	0.8	0.0	7.2	1.5
		20	8	3	1	1	7	0.6	0.1	9.5	5.9
		21	3	7	1	2	-1	0.5	0.9	4.1	-4.9
		22	5	6	2	0	1	0.4	0.0	5.5	-2.0
		23	1	10	1	1	-7	0.6	1.0	1.6	-9.8
		27	0	11	0	2	-9	0.0	0.4	1.4	-10.2
12	0	0	10*	0	3	0	13*	0.9	0.0	12.3*	11.6*
		1	7	3	2	1	7	0.8	1.0	7.9	2.9
		2	7	3	2	1	7	0.2	0.1	8.9	4.8
		3	8	3	1	1	7	0.2	0.7	8.5	4.0
		4	1	12	0	0	-11	0.0	0.0	1.0	-11.0
		5	7	5	0	1	3	0.0	0.6	7.4	1.8
		6	4	6	0	3	1	0.0	0.7	5.1	-2.9
		7	9	3	1	0	7	0.9	0.0	9.9	6.7
		10	4	4	5	0	5	0.5	0.0	5.8	-1.3
		20	6	4	1	2	5	0.6	0.4	8.2	3.5
		21	4	6	1	2	1	0.5	0.4	5.7	-1.6
		22	5	7	1	0	-1	0.2	0.0	5.2	-2.7
		23	1	11	1	0	-9	0.6	0.0	1.4	-9.8
		27	1	7	0	5	-1	0.0	1.0	3.4	-5.9
13	21	21	11*	1	1	0	11*	0.6	0.0	11.4*	10.2*
		0	8	1	4	0	11*	0.7	0.0	9.9	6.8
		1	5	7	1	0	-1	0.3	0.0	5.3	-2.5
		2	8	3	1	1	7	0.4	0.2	9.4	5.7
		3	7	4	1	1	5	0.9	0.1	8.8	4.6
		4	1	10	1	1	-7	0.2	0.9	1.3	-10.4
		5	6	7	0	0	-1	0.0	0.0	6.0	-1.0
		6	1	9	0	3	-5	0.0	0.7	2.3	-7.3
		7	5	7	1	0	-1	0.6	0.0	5.6	-1.8
		10	5	2	6	0	9	0.8	0.0	7.5	2.0
		20	8	2	1	2	9	0.7	0.7	9.8	6.6
		22	7	4	2	0	5	0.7	0.0	8.1	3.3
		23	1	10	1	1	-7	0.7	0.9	1.8	-9.4
		27	2	8	0	3	-3	0.0	0.8	3.1	-6.8
14	21	20	10*	1	1	1	11*	0.6	0.0	11.6*	10.1*
	21	21	9	3	1	0	7	0.5	0.0	9.5	6.0
		0	7	1	5	0	11*	0.9	0.0	9.3	5.7
		1	7	2	4	0	9	0.9	0.0	8.9	4.9
		2	6	3	4	0	7	0.9	0.0	8.2	3.3
		3	5	5	2	1	3	0.7	0.9	6.6	0.2
		4	1	12	0	0	-11	0.0	0.0	1.0	-11.0
		5	6	7	0	0	-1	0.0	0.0	6.0	-1.0
		6	1	9	0	3	-5	0.0	1.0	2.1	-3.8
		7	4	6	2	1	1	0.1	0.3	4.4	-4.2
		10	5	2	6	0	9	0.6	0.0	7.9	2.9
		22	5	4	2	2	5	0.9	0.4	7.7	2.4
		23	2	8	1	2	-3	0.6	0.9	3.1	-6.8
		27	3	8	0	2	-3	0.0	0.1	4.7	-3.6
15	21	21	10*	1	2	0	11	0.5	0.0	11.0	9.1
	21	22	10*	1	2	0	11	0.9	0.0	11.3*	9.7*
		0	7	4	2	0	5	0.5	0.0	7.9	2.9
		1	4	8	1	0	-3	0.6	0.0	4.6	-3.8
		2	6	5	0	2	3	0.0	0.5	7.2	1.5
		3	4	7	0	2	-1	0.0	0.4	5.4	-2.3

		4	4	7	0	2	-1	0.0	0.1	5.8	-1.2
		5	3	8	0	2	-3	0.0	0.5	4.4	-4.2
		6	1	8	0	4	-3	0.0	0.8	3.1	-2.7
		7	5	7	0	1	-1	0.0	0.1	5.9	-1.1
		10	8	0	5	0	13*	0.5	0.0	10.0	7.0
		20	6	3	1	3	7	0.6	0.6	8.6	4.3
		23	3	2	1	7	9	0.6	0.9	4.9	-3.1
		27	0	10	0	3	-7	0.0	1.0	0.7	-11.7
16	21	21	11*	1	1	0	11*	0.6	0.0	11.6*	10.1*
		0	6	3	4	0	7	0.8	0.0	8.2	3.4
		1	5	7	1	0	-1	0.3	0.0	5.3	-2.4
		2	7	5	0	1	3	0.0	0.0	8.0	3.0
		3	6	5	0	2	3	0.0	1.0	6.9	0.9
		4	2	7	2	2	-1	0.2	1.0	2.6	-7.9
		5	4	7	0	2	-1	0.0	0.1	5.9	-1.2
		6	1	9	0	3	-5	0.0	0.6	2.8	-7.3
		7	5	6	1	1	1	0.8	0.9	5.9	-1.2
		10	6	1	6	0	11*	0.9	0.0	8.5	4.0
		20	8	2	1	2	9	0.6	0.6	9.9	6.7
		22	9	3	1	0	7	0.9	0.0	9.9	6.3
		23	2	8	1	2	-3	0.7	0.9	2.9	-7.3
		27	1	9	0	3	-5	0.0	0.8	2.5	-9.0
17	0	7	9*	3	1	0	7	0.4	0.0	9.4	5.9
	0	0	8	0	5	0	13*	0.8	0.0	11.5*	10.0*
		1	8	3	2	0	7	0.7	0.0	8.8	4.7
		2	8	2	3	0	9	0.8	0.0	9.5	4.0
		3	7	2	1	3	9	0.3	0.4	9.6	6.1
		4	5	7	1	0	-1	0.2	0.0	5.2	-2.6
		5	7	5	0	1	3	0.0	0.7	7.2	1.6
		6	4	4	0	5	5	0.0	1.0	6.3	-0.4
		10	4	3	6	0	7	0.7	0.0	7.3	1.6
		20	4	7	1	1	-1	0.5	0.2	5.3	-2.4
		21	2	10	1	0	-7	0.4	0.0	2.4	-8.1
		22	5	5	2	1	3	0.3	0.0	6.4	-0.1
		23	1	9	1	2	-5	0.5	1.0	1.6	-9.8
		27	0	12	0	1	-11	0.0	0.3	0.2	-12.5
18	0	0	9*	0	4	0	13*	0.8	0.0	11.9*	10.8*
	0	7	9*	3	1	0	7	0.6	0.0	9.6	6.2
		1	7	4	2	0	5	0.7	0.0	7.9	2.8
		2	4	3	3	3	7	0.6	0.9	6.5	-0.1
		3	6	3	1	3	7	0.2	1.0	7.3	1.5
		4	7	4	0	2	5	0.0	0.3	8.5	4.0
		5	6	5	0	2	3	0.0	0.3	7.6	2.3
		6	8	2	0	3	9	0.0	0.7	10.1	7.3
		10	5	3	5	0	7	0.6	0.0	7.6	2.2
		20	2	8	1	2	-3	0.4	0.3	4.1	-4.8
		21	2	10	1	0	-7	0.4	0.0	2.4	-8.2
		22	4	7	1	1	-1	0.4	0.7	4.7	-3.6
		23	2	8	1	2	-3	0.5	1.0	2.7	-7.6
		27	0	11	0	2	-9	0.0	1.0	0.1	-12.8
19	0	0	10*	0	3	0	13*	1.0	0.0	12.4*	11.8*
	0	1	10*	2	1	0	9	0.8	0.0	10.8	8.6
		2	7	2	3	1	9	0.7	0.0	8.9	4.8
		3	9	1	2	1	11	0.3	1.0	9.6	6.3
		4	5	7	0	1	-1	0.0	0.6	5.4	-2.1
		5	1	10	0	2	-7	0.0	0.4	2.3	-8.3
		6	4	4	0	5	5	0.0	0.9	6.5	0.0
		7	7	3	1	2	7	0.7	0.3	9.4	5.7
		10	4	3	6	0	7	0.9	0.0	6.7	0.4
		20	2	8	1	2	-3	0.6	0.1	4.5	-4.1
		21	1	9	1	2	-5	0.6	0.8	2.0	-9.0
		22	4	6	1	2	1	0.1	0.4	5.4	-2.2
		23	1	8	1	3	-3	0.6	1.0	2.2	-9.5
		27	4	6	0	3	1	0.0	0.9	4.9	-3.2
20	0	1	11*	2	0	0	9	0.0	0.0	11.0	9.0
	0	0	10	0	3	0	13*	0.9	0.0	12.4*	11.9*

		2	9	3	1	0	7	0.7	0.0	9.7	6.4
		3	9	2	2	0	9	0.3	0.0	9.4	6.1
		4	4	7	1	1	-1	0.1	0.3	4.7	-3.5
		5	1	10	0	2	-7	0.0	0.4	2.3	-2.5
		6	3	7	0	3	-1	0.0	0.7	4.7	-3.6
		7	9	3	0	1	7	0.0	0.7	9.3	5.5
		10	4	4	5	0	5	0.5	0.0	5.9	-1.2
		20	4	7	1	1	-1	0.7	0.0	5.7	-1.6
		21	1	9	1	2	-5	0.6	0.6	2.4	-8.1
		22	6	6	1	0	1	0.5	0.0	6.5	0.0
		23	0	10	1	2	-7	0.6	0.8	1.1	-10.8
		27	4	5	0	4	3	0.0	1.0	5.3	-1.5
21	0	0	11*	0	2	0	13*	0.8	0.0	12.5*	12.0*
	0	1	11*	2	0	0	9	0.0	0.0	11.0	9.0
		2	10	3	0	0	7	0.0	0.0	10.0	7.0
		3	8	2	1	2	9	0.3	0.3	9.9	6.8
		4	4	8	0	1	-3	0.0	0.8	4.2	-4.6
		5	1	10	0	2	-7	0.0	0.7	2.1	-8.9
		6	3	7	0	3	-1	0.0	0.9	4.2	-4.6
		7	9	4	0	1	5	0.0	0.9	8.1	3.2
		10	4	4	5	0	5	0.5	0.0	5.7	-1.6
		20	5	6	1	1	1	0.7	0.0	6.7	0.5
		21	2	8	1	2	-3	0.6	0.3	4.1	-4.9
		22	4	8	1	0	-3	0.5	0.0	4.5	-3.9
		23	0	10	1	2	-7	0.6	0.9	1.1	-10.8
		27	5	4	0	4	5	0.0	1.0	6.9	0.8
22	1	0	10*	0	3	0	13*	1.0	0.0	12.5*	11.9*
	1	1	10*	0	2	1	13*	0.0	0.0	11.0	9.1
		2	9	4	0	0	5	0.0	0.0	9.0	5.0
		3	9	3	1	0	7	0.3	0.0	9.3	5.6
		4	2	11	0	0	-9	0.0	0.0	2.0	-9.0
		5	1	11	0	1	-9	0.0	0.2	1.8	-9.4
		6	3	9	0	1	-5	0.0	0.2	3.8	-5.4
		7	5	6	2	0	1	1.0	0.0	6.9	0.9
		10	4	3	6	0	7	1.0	0.0	6.2	-0.7
		20	8	4	1	0	5	0.8	0.0	8.8	4.6
		21	5	6	1	1	1	0.6	0.7	5.9	-1.1
		22	3	9	1	0	-5	0.0	0.0	3.0	-6.9
		23	1	9	1	2	-5	0.7	0.8	2.1	-8.9
		27	8	3	0	2	7	0.0	1.0	8.7	4.4
23	21	21	11*	1	1	0	11*	0.7	0.0	11.7*	10.5*
		0	9	1	3	0	11*	1.0	0.0	11.0	9.0
		1	6	5	2	0	3	0.9	0.0	7.3	1.5
		2	8	4	1	0	5	0.7	0.0	8.7	4.3
		3	8	3	1	1	7	0.9	0.7	9.2	5.4
		4	3	9	1	0	-5	0.6	0.0	3.6	-5.7
		5	1	10	0	2	-7	0.0	1.0	1.7	-9.6
		6	2	9	0	2	-5	0.0	0.4	3.3	-6.4
		7	4	8	1	0	-3	0.1	0.0	4.1	-4.8
		10	2	4	7	0	5	0.4	0.0	3.3	-6.4
		20	8	2	1	2	9	0.9	0.6	10.0	6.9
		22	6	5	1	1	3	0.7	1.0	6.7	0.5
		23	0	9	1	3	-5	0.8	1.0	1.4	-10.2
		27	5	3	0	5	7	0.0	0.6	9.0	5.0
24	0	0	11*	0	2	0	13*	0.8	0.0	12.6*	12.2*
	0	1	11*	2	0	0	9	0.0	0.0	11.0	9.0
		2	9	4	0	0	5	0.0	0.0	9.0	5.0
		3	9	1	1	2	11	0.2	0.9	9.7	6.4
		4	3	9	0	1	-5	0.0	1.0	3.0	-7.0
		5	1	10	0	2	-7	0.0	0.7	2.1	-8.8
		6	3	7	0	3	-1	0.0	0.9	4.0	-5.0
		7	8	4	0	1	5	0.0	0.1	8.9	4.3
		10	4	4	5	0	5	0.7	0.0	5.9	-1.2
		20	7	5	1	0	3	0.8	0.0	7.8	2.5
		21	2	8	1	2	-3	0.6	0.3	4.2	-4.6
		22	4	8	1	0	-3	0.3	0.0	4.3	-4.4

		23	0	10	1	2	-7	0.6	0.8	1.1	-10.9
		27	5	5	0	3	3	0.0	0.4	7.5	1.9
25	1	0	11*	0	2	0	13*	0.8	0.0	12.5*	12.1*
	1	1	11*	2	0	0	9	0.0	0.0	11.0	9.0
		2	8	5	0	0	3	0.0	0.0	8.0	3.0
		3	9	3	1	0	7	0.3	0.0	9.3	5.6
		4	2	11	0	0	-9	0.0	0.0	2.0	-9.0
		5	1	10	0	2	-7	0.0	1.0	1.8	-9.3
		6	3	9	0	1	-5	0.0	0.2	3.8	-5.4
		7	8	5	0	0	3	0.0	0.0	8.0	3.0
		10	4	4	5	0	5	0.9	0.0	6.0	-1.0
		20	8	4	1	0	5	0.8	0.0	8.8	4.6
		21	4	7	1	1	-1	0.6	0.0	5.6	-1.8
		22	4	8	1	0	-3	0.1	0.0	4.1	-4.8
		23	0	10	1	2	-7	0.7	0.8	1.1	-10.9
		27	9	4	0	0	5	0.0	0.0	9.0	5.0
26	21	21	10*	1	2	0	11	0.7	0.0	11.2*	9.4*
		0	8	2	3	0	9	1.0	0.0	10.4	7.8
		1	6	5	2	0	3	0.1	0.0	6.1	-0.8
		2	9	4	0	0	5	0.0	0.0	9.0	5.0
		3	9	4	0	0	5	0.0	0.0	9.0	5.0
		4	3	8	1	1	-3	0.4	0.7	3.7	-5.6
		5	1	11	0	1	-9	0.0	0.4	1.6	-9.8
		6	2	8	0	3	-2	0.0	0.9	3.2	-6.5
		7	3	8	1	1	-2	0.5	0.9	3.6	-5.9
		10	6	0	7	0	13*	0.9	0.0	8.4	3.8
		20	8	2	1	2	9	0.9	0.5	10.1	7.1
		22	6	5	2	0	3	0.9	0.0	7.4	1.8
		23	0	10	1	2	-7	0.8	0.7	1.5	-10.0
		27	3	6	0	4	1	0.0	0.7	5.9	-1.3

PLACEMENT OF IDENTIFIED SETS

	YVOTE	CVOTE	NVOTE	BVOTE
UNIQUE 1ST PLACE	11	12	19	19
TIED 1ST PLACE	9	6	0	0
OTHER	6	8	7	7

DISCRIMINATION SUCCESS BY HYPERPLANE

HYPERPLANE		POINTS IN THE FIRST SET				POINTS IN THE SECOND SET				
#SET1	SET2	YVOTE	NVOTE	OVOTE	CVOTE	YXT	YVOTE	NVOTE	OVOTE	CVOTE
1	0	6	12	0	0	100.				
2	0	27	12	0	0	100.				
6	1	6	4	0	0	100.				
7	1	27	3	0	0	75.				
8	21	6	9	0	0	100.				
9	21	27	9	0	0	100.				
26	2	3	0	1	0	0.	1	0	0	100.
27	2	4	1	0	0	100.				
28	2	6	1	0	0	100.				
29	2	27	1	0	0	100.				
35	21	2	9	0	0	100.				
36	21	3	8	1	0	89.	1	0	0	100.
37	21	4	9	0	0	100.				
42	21	7	9	0	0	100.				
43	21	22	5	1	3	56.	1	0	0	100.
44	21	5	9	0	0	100.				
45	21	23	9	0	0	100.				
46	1	7	0	1	3	0.	1	0	0	100.
47	1	22	4	0	0	100.				
48	1	5	4	0	0	100.				
49	1	23	4	0	0	100.				
54	0	7	12	0	0	100.				
55	0	22	12	0	0	100.				
56	0	5	12	0	0	100.				
57	0	23	12	0	0	100.				
58	2	20	1	0	0	100.				

61	1	2	3	0	1	0	75.	0	0	1	0	0.
62	1	3	4	0	0	0	100.					
63	1	4	4	0	0	0	100.					
64	2	10	1	0	0	0	100.					
67	0	2	7	0	5	0	58.	0	0	1	0	0.
68	0	3	0	0	12	0	0.					
69	0	4	7	0	5	0	58.					
70	21	1	8	1	0	0	89.	1	0	0	0	100.
71	21	20	8	1	0	0	89.	1	0	0	0	100.
72	1	20	4	0	0	0	100.					
73	21	10	0	0	9	0	0.					
75	0	20	12	0	0	0	100.					
76	21	0	0	9	0	0	0.	9	0	0	0	100.
77	1	10	4	0	0	0	100.					
78	1	0	0	1	3	0	0.	1	0	5	0	17.
79	0	10	0	0	12	0	0.					
80	2	7	0	1	0	0	0.	1	0	0	0	100.
81	2	22	1	0	0	0	100.					
82	2	5	1	0	0	0	100.					
83	2	23	1	0	0	0	100.					

TIME USED: 8.54 SECONDS, COMPLETED AT 16.75 HOURS
 ENTER 0,1 AS THERE ARENT,ARE MORE CASES

0
 =====
 FORTRAN STOP

* EXIT
 CASEY Job terminated at 22-MAY-1987 16:45:08.61

Accounting information:
 Buffered I/O count: 54 Peak working set size: 1206
 Direct I/O count: 110 Peak page file size: 1783
 Page faults: 1351 Mounted volumes: 0
 Charged CPU time: 0 00:00:07.23 Elapsed time: 0 00:00:13.89
 VAX4>

VAX4> RUN EFBLBN

EFAGHY (EMPIRICAL FLUTTER ANALYSIS BY GALACTIC HYPERPLANES)
RUN ON 10/14/87 AT 14.63 HOURS ON TSHARE

ENTER NUMBER & NAMES OF DATA FILES, WORDS/RECORD, Y/N TO CLEARFILES
1 755777B7 20 7Y7

%FOR-F-INPCONERR, input conversion error

unit 5 file SYS\$INPUT:;

user PC 00009BA1

%TRACE-F-TRACEBACK, symbolic stack dump follows

module name	routine name	line	rel PC	abs f
			000140C2	000140
			000140EF	000140
			00010D68	00010D
EFBLBN\$MAIN	EFBLBN\$MAIN	117	000001A1	00009E

VAX4> RUN EFBLBN

EFAGHY (EMPIRICAL FLUTTER ANALYSIS BY GALACTIC HYPERPLANES)
RUN ON 10/14/87 AT 14.64 HOURS ON TSHARE

ENTER NUMBER & NAMES OF DATA FILES, WORDS/RECORD, Y/N TO CLEARFILES
1 '55777B' 20 'Y'

NAME GALACTIC HYPERPLANE FILE

'FLPLANES'

ENTER 0,1 TO NOTPERTURB, PERTURB INPUT DATA FILES

0

NAME OUTPUT FILE FOR SET ASSIGNMENTS, ' ' TO PRINT

7 7

%FOR-F-INPCONERR, input conversion error

unit 5 file SYS\$INPUT:;

user PC 00009DED

%TRACE-F-TRACEBACK, symbolic stack dump follows

module name	routine name	line	rel PC	abs F
			000140C2	000140
			000140EF	000140
			00010D68	00010D
EFBLBN\$MAIN	EFBLBN\$MAIN	154	000003ED	00009E

VAX4> RUN EFBLBN

EFAGHY (EMPIRICAL FLUTTER ANALYSIS BY GALACTIC HYPERPLANES)
RUN ON 10/14/87 AT 14.66 HOURS ON TSHARE

ENTER NUMBER & NAMES OF DATA FILES, WORDS/RECORD, Y/N TO CLEARFILES
1 '55777B' 20 'Y'

NAME GALACTIC HYPERPLANE FILE

'FLPLANES'


```

'PLPLANES'
ENTER 0,1 TO NOTPERTURB,PERTURB INPUT DATA FILES
0
NAME OUTPUT FILE FOR SET ASSIGNMENTS , ' ' TO PRINT
'
ENTER N (LE 14) THEN ID OF N SETS TO BE EXCLUDED
0 0
ENTER N, THEN N PAIRS OF IDS TO BE LABELED AS THE FIRST
0 0
ENTER N THEN THE N HYPERPLANES TO BE USED(0 0 FOR ALL)
0 0
ENTER 0 OR 1 TO NOTWRITE OR WRITE THE FOLLOWING OUTPUT OPTIONS
HYP,PTS,VOT,ASS,ASP,ASC,HPT
0111111
ENTER N (LE 20) THEN N RECORD NUMBERS FOR HYPERPLANE/POINT OUTPUT
2
1 2 3

```

P VOTE CONFIDENCE LEVEL = 0.5000000

RELIABILITY INDEX PER SET

ID	0	1	2	3	4	5	6	7	10	20	21	22	23	27
X	0.71	0.73	0.77	0.78	0.71	0.31	0.08	0.77	0.59	0.79	0.73	0.67	0.57	0.17

PT ACTID CANID YVOT NVOT OVOT GVOT CVOT MX%O MX%G WVOTE BVOTE PVOTE

DATA FROM FILE 55777B

POINT NUMBER	1				
432.0000	67.50000	904.3000	19.50000	2.790000	
1.321000	49.91700	0.6860000	10.36900	1.700000	
0.6650000	1.141000	0.1300000	937.0000	4.629000	
1.770000	0.0000000E+00	0.0000000E+00	21.00000	21.00000	

HYPERPLANE VOTES FOR POINT #			432, RECORD #	DISTANCES NORMAL TO PLANES			1
HYPL#	SETID					SETID	
1	6	1	27.21380	27.21381E	61.63589	0	
2	27	1	24.01917	24.020011	56.53007	0	
3	27	1	-1.04922	-0.85887	-0.40235E	6	
4	6	1	-0.18576	-0.18574E	-0.11681	10	
5	27	1	0.10979	0.10980E	0.13346	10	
6	6	1	-0.14940	-0.14234E	0.06251	1	
7	27	1	-2.28858	-2.28303E	-2.23948	1	
8	6	1	0.07854	0.15722E	0.26433	21	
9	27	1	-2.95282	-2.94929E	-2.92339	21	
10	6	1	-1.98267	-1.94685	-1.92326E	20	
11	27	1	-2.03761	-1.96018	-1.92418E	20	
12	22	0.33629E	0.52278	0.57952E		7	
13	5	1	-1.06194	-1.04936E	-1.02470	7	
14	6	1	-1.28627	-1.25879E	-0.97135	7	
15	27	1	-1.46843	-1.42749	-1.37155E	7	

16	23	1	-2.51894	-2.31391	-2.30384		7
17	5	1	0.03803		0.03929	0.57164	22
18	6	1	-0.61770		-0.50018	-0.43982	22
19	27	-0.95190	-0.93183		-0.91507		22
20	23	1	-1.69360		-1.67326	-1.42078	22
21	6	-0.95190	-0.59412		-0.58419		5
22	27	-0.95190	-0.93183		-0.58419		5
23	23	1	-2.68281	-2.49879	-2.25038		5
24	23	1	-2.10593	-1.77685	-1.50922		6
25	23	1	0.73624		0.93183	0.95190	27
26	3	1	-0.21974		-0.21910	-0.21490	2
27	4	-1.06349	-0.99242		-0.97353		2
28	6	1	-2.15026		-2.09786	-2.08703	2
29	27	1	-0.49929		-0.42550	-0.28202	2
30	4	-1.13402	-1.07920		-1.06043		3
31	6	1	-1.99838		-1.88772	-1.79691	0
32	27	1	-2.60251		-2.55897	-2.49017	3
33	6	1	1.10985	1.11110	1.11351		4
34	27	1	-0.61296	-0.40740	-0.36990		4
35	2	1	-0.39357		-0.39357	-0.38708	21
36	3	1	0.13106		0.13107	0.15767	21
37	4	1	0.13857		0.13857	0.14462	21
38	7	1	0.12849		0.12850	0.18533	20
39	22	0.33293	0.34155		0.36176		20
40	5	1	-0.10342		-0.09257	0.40926	20
41	23	1	-0.54755	-0.53242	-0.47550		20
42	7	1	0.37558		0.40575	0.43649	21
43	22	0.20672	0.20682		0.20925		21
44	5	1	0.13619		0.40553	0.55748	21
45	23	1	-2.27999		-2.25574	-2.02524	21
46	7	1	-0.02355	-0.02090	-0.01963		1
47	22	0.30729	0.32760		0.32808		1
48	5	1	-0.13876		-0.13876	0.28776	1

49	23		0.20178	0.21807	0.27575		1
50	7		-0.23324		-0.20613	-0.10642	10
51	22		0.10292	0.10609	0.11091		10
52	5		-1.97558		-1.97556	17.82939	10
53	23		0.27362	0.36437	0.55480		10
54	7	0.79066	0.82410		0.92761		0
55	22		21.73051		21.73053	50.19141	0
56	5		28.22050		28.22051	65.47410	0
57	23	0.25481	0.25994		0.25999		0
58	20	-0.52030	-0.50071		-0.50071		2
59	20	-0.08324	-0.05049		-0.05049		3
60	4	0.00779	0.01297		0.01298		20
61	2	0.47075	0.51116		0.54830		1
62	3	0.28201	0.29632		0.29656		1
63	4		0.30130		0.30130	0.30340	1
64	10	-0.18290	-0.16497		-0.16340		2
65	10	0.14991	0.24728		0.24733		3
66	4		0.38866	0.48291	0.49949		10
67	2	0.09612	0.15425		0.22160		0
68	3	0.07739	0.12111		0.25828		0
69	4		0.19506		0.19941	0.20029	0
70	1		-0.17130		-0.17130	-0.15081	21
71	20		-0.07224		-0.04246	-0.03696	21
72	20	0.23260	0.27950		0.27950		1
73	10		-0.43299	-0.30725	-0.24224		21
74	10		-0.41106	-0.22954	-0.18058		20
75	20		23.07658		24.38633	71.27054	0
76	0	-71.10430	-20.16585		-20.16562		21
77	10	0.43799	0.49614		0.52800		1
78	0		0.23361	0.23428	0.25680		1
79	10		-0.23210	-0.03140	0.02505		0
80	7		0.07367		0.07367	0.10607	2
81	22		0.39447		0.41195	0.44982	2

82	5	1	0.74082		0.81886E	0.85541	2					
83	23	1	-2.33588	-2.06975	-1.99848E		2					
84	7	1	0.15241		0.16586E	0.38493	3					
85	22	-0.26118E	-0.15288		-0.13143E		3					
86	5	-2.18022E	-2.17010		-2.14888E		3					
87	23	1	-2.15842	-1.89041	-1.85990E		3					
88	4	0.62747E	0.63751		0.63769E		7					
89	4	1	0.67205		0.67987E	0.75043	22					
90	5	1	1.85836		1.86168E	2.26662	4					
91	23	1	-2.45007		-2.26319E	-2.24415	4					
432	21	21	10*	2	1	0	9	0.7	0.0	10.7*	8.3*	8.2*
		0	7	4	2	0	5	1.0	0.0	8.8	4.5	5.1
		1	4	6	2	1	1	0.7	0.8	4.9	-3.1	-0.7
		2	8	4	0	1	5	0.0	0.2	8.8	4.6	2.5
		3	5	7	0	1	-1	0.0	0.1	5.9	-1.2	-1.8
		4	6	4	1	2	5	0.1	0.7	7.3	1.7	3.3
		5	1	11	0	1	-9	0.0	0.6	1.4	-10.1	-2.9
		6	1	8	0	4	-3	0.0	0.7	2.9	-7.2	-0.5
		7	3	7	1	2	-1	0.3	0.6	4.7	-3.6	-3.6
		10	7	0	6	0	13*	0.9	0.0	9.3	5.7	6.6
		20	5	4	1	3	5	0.8	0.8	7.3	1.6	1.9
		22	9	3	1	0	7	0.6	0.0	9.6	6.2	5.9
		23	1	4	1	7	5	0.7	1.0	4.6	-3.7	-2.4
		27	3	6	0	4	1	0.0	0.8	4.8	-3.5	-0.3

POINT NUMBER	2			
434.0000	67.50000	910.4000	17.66000	2.070000
1.321000	60.99600	0.9560000	12.16400	7.920000
0.9080000	1.282000	0.1000000	1242.000	5.272000
1.990000	0.0000000E+00	0.0000000E+00	1.000000	1.000000

HYPERPLANE VOTES FOR POINT #		434, RECORD #		2	
HYPL#	SETID	DISTANCES NORMAL TO PLANES		SETID	
1	6	1	27.21380	27.21381E	62.11111
2	27	1	24.01917	24.02001E	57.06602
3	27	1	-1.04922	-0.96216	-0.40235E
4	6	1	-0.18576	-0.18574E	-0.08219
5	27	1	0.10979	0.10980E	0.14937
6	6	1	-0.14940	-0.14234E	-0.01629
7	27	1	-2.28858	-2.28303E	-2.25563
8	6	1	0.07854	0.13383	0.15722E
9	27	1	-2.95282	-2.95179	-2.94939E
10	6	1	-1.98267	-1.92326E	-1.89979
11	27	1	-2.03761	-1.92418E	-1.89643
12	6	1	0.50000	0.50000E	0.50000

12	22		0.52278		0.579521	0.58783	7
13	5	-1.173451	-1.06194		-1.049361		7
14	6	-1.307581	-1.28627		-1.258791		7
15	27	-1.479431	-1.46843		-1.371551		7
16	23		1 -2.51894		-2.303841	-2.19560	7
17	5		1 0.03803		0.039291	0.54908	22
18	6		1 -0.41770		-0.500181	-0.30800	22
19	27	-0.984711	-0.93183		-0.915071		22
20	23		1 -1.69360		-1.673261	-1.52027	22
21	6	-0.984711	-0.59412		-0.584191		5
22	27	-0.984711	-0.93183		-0.584191		5
23	23		1 -2.68281	-2.37346	-2.250381		5
24	23		1 -2.10593	-1.64220	-1.509221		6
25	23		1 0.73684		0.931831	0.98471	27
26	3		1 -0.21974		-0.218101	-0.21729	2
27	4		1 -0.99242		-0.973531	-0.90537	2
28	6		1 -2.15086		-2.097861	-2.00979	2
29	27		1 -0.49928		-0.425501	-0.37283	2
30	4		1 -1.07920		-1.060431	-0.99663	3
31	6		1 -1.99838		-1.887721	-1.73158	3
32	27		1 -2.60251		-2.558971	-2.55111	3
33	6	1.009351	1.10985		1.113511		4
34	27		1 -0.61296	-0.57065	-0.369901		4
35	2	-0.420451	-0.39357		-0.393571		21
36	3	0.111831	0.13106		0.131071		21
37	4	0.114741	0.13857		0.138571		21
38	7	0.033671	0.12849		0.128501		20
39	22		1 0.34155		0.361761	0.38486	20
40	5		1 -0.10342		-0.092571	0.48003	20
41	23		1 -0.54755		-0.475501	-0.31534	20
42	7	0.345741	0.37558		0.405751		21
43	22		1 0.20682		0.209251	0.21063	21
44	5		1 0.13619	0.34883	0.405531		21
45	22		1 0.13619	0.34883	0.405531		21

45	23		J	-2.27999		-2.25574	-1.87820	21
46	7		[-0.02355	-0.01999	-0.01963		1
47	22		J	0.32760		0.32808	0.34796	1
48	5		J	-0.13876		-0.13876	0.26276	1
49	23		J	0.20178		0.27575	0.51205	1
50	7		[-0.23324	-0.21929	-0.20613		10
51	22	0.10229	[0.10292		0.11091		10
52	5		J	-1.97558		-1.97556	19.06431	10
53	23		[0.27362	0.37969	0.55480		10
54	7		[0.82410		0.92761	1.01779	0
55	22		[21.73051		21.73053	51.02850	0
56	5		J	28.22050		28.22051	66.49910	0
57	23		J	0.25994		0.25999	0.26782	0
58	20		J	-0.50071		-0.50071	-0.45170	2
59	20		[-0.05049		-0.05049	-0.04300	3
60	4		J	0.01297		0.01298	0.01303	20
61	2		[0.51116	0.54682	0.54830		1
62	3		[0.29632		0.29656	0.30276	1
63	4		[0.30130		0.30130	0.31284	1
64	10		[-0.16497		-0.16340	-0.15443	2
65	10		[0.24728		0.24733	0.28322	3
66	4		[0.38866		0.49949	0.52118	10
67	2		[0.15425	0.20174	0.22160		0
68	3		[0.12111	0.19805	0.25828		0
69	4		[0.19506		0.19941	0.20218	0
70	1	-0.21734	[-0.17130		-0.17130		21
71	20		[-0.07224	-0.06846	-0.04246		21
72	20		J	0.27950		0.27950	0.31149	1
73	10		[-0.43299	-0.32170	-0.24224		21
74	10		[-0.41106	-0.24596	-0.18058		20
75	20		[23.07658		24.38633	69.00265	0
76	0	-69.88106	[-20.16585		-20.16562		21
77	10		[0.49614		0.52800	0.56654	1
78	0		[0.00000	0.00000	0.00000		

76	0	1	0.23361	0.23870	0.236801	1						
79	10	1	-0.23210	-0.03948	0.025051	0						
80	7	1	0.07367		0.073671	0.08325	2					
81	22	1	0.39447		0.411951	0.51808	2					
82	5	1	0.74082		0.818861	1.09217	2					
83	23	1	-2.33588		-1.998481	-1.92019	2					
84	7	1	0.15241		0.165861	0.21590	3					
85	22	1	-0.15288		-0.131431	-0.11144	3					
86	5	1	-2.17010		-2.148881	-2.06792	3					
87	23	1	-2.15842		-1.859901	-1.74849	3					
88	4	1	0.63751		0.637691	0.76324	7					
89	4	1	0.67205		0.679871	0.79468	22					
90	5	1	1.85836		1.861681	2.26121	4					
91	23	1	-2.45007		-2.263191	-2.12894	4					
434	1	2	11*	0	2	0	13*	0.3	0.0	11.3	9.7	9.1
	1	1	10	0	3	0	13*	1.0	0.0	12.1*	11.2*	9.3*
		0	9	0	4	0	13*	0.2	0.0	11.8	10.6	8.5
		3	10	2	1	0	9	0.4	0.0	10.4	7.9	7.8
		4	3	9	0	1	-5	0.0	0.2	3.2	-6.7	-3.9
		5	1	10	0	2	-7	0.0	0.8	1.9	-9.1	-2.6
		6	3	7	0	3	-1	0.0	0.9	4.2	-4.6	-0.1
		7	5	6	2	0	1	0.5	0.0	5.6	-1.9	-0.1
		10	4	4	5	0	5	0.5	0.0	5.8	-1.3	-0.1
		20	6	5	2	0	3	0.9	0.0	7.6	2.2	2.5
		21	2	6	2	3	1	0.6	0.7	4.5	-4.0	-2.9
		22	5	8	0	0	-3	0.0	0.0	5.0	-3.0	-0.9
		23	0	10	1	2	-7	0.6	0.8	1.1	-10.7	-7.0
		27	4	6	0	3	1	0.0	0.3	6.4	-0.2	0.1

POINT NUMBER	3			
436.0000	67.50000	915.0000	15.45000	2.430000
1.321000	61.23700	0.8250000	10.69200	10.98000
0.7810000	1.529000	0.1400000	1091.000	4.630000
1.770000	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00

HYPERPLANE VOTES FOR POINT #		436, RECORD #		3	
HYPL#	SETID	DISTANCES NORMAL TO PLANES		SETID	
1	6	1	27.21380	27.213811	61.30742
2	27	1	24.01917	24.020011	56.28197
3	27	1	-1.04922	-0.74537	-0.402351
4	6	1	-0.18576	-0.185741	-0.11602
5	27	1	0.10979	0.109801	0.20062
6	6	1	-0.14940	-0.142341	-0.00193
7	27	1	-2.28858	-2.283031	-2.28075
8	6	1	0.07854	0.10009	0.157221

9	27	-2.984261	-2.95282		-2.949391		21
10	6		-1.98267	-1.93160	-1.923261		20
11	27		-2.03761	-1.93980	-1.924181		20
12	22		0.52278		0.579521	0.64997	7
13	5	-1.148501	-1.06194		-1.049361		7
14	6	-1.329211	-1.28627		-1.258791		7
15	27		-1.46843	-1.42111	-1.371551		7
16	23		-2.51894		-2.303841	-2.23848	7
17	5		0.03803		0.039291	0.53499	22
18	6		-0.61770		-0.500181	-0.47680	22
19	27	-1.028181	-0.93183		-0.915071		22
20	23		-1.69360		-1.673261	-1.38210	22
21	6	-1.028181	-0.59412		-0.584191		5
22	27	-1.028181	-0.93183		-0.584191		5
23	23		-2.68281	-2.41077	-2.250381		5
24	23		-2.10593	-1.62406	-1.509221		6
25	23		0.70684		0.931831	1.02818	27
26	3		-0.21974	-0.21829	-0.218101		2
27	4		-0.99242		-0.973531	-0.94390	2
28	6		-2.15086	-2.13301	-2.097861		2
29	27		-0.49928		-0.425501	-0.16251	2
30	4		-1.07920		-1.060431	-1.01920	3
31	6		-1.99838		-1.887721	-1.84289	3
32	27		-2.60251		-2.558971	-2.51132	3
33	6	1.042611	1.10985		1.113511		4
34	27		-0.61296	-0.37576	-0.369901		4
35	2	-0.430801	-0.39357		-0.393571		21
36	3	0.109451	0.13106		0.131071		21
37	4	0.097451	0.13857		0.138571		21
38	7	-0.061271	0.12849		0.128501		20
39	22	0.275511	0.34155		0.361761		20
40	5		-0.10342		-0.092571	0.41357	20
41	23		-0.54755	-0.52170	-0.475501		20

42	7	0.286881	0.37558		0.405751		21
43	22	0.203911	0.20682		0.209251		21
44	5		0.13619	0.21161	0.405531		21
45	23		-2.27999		-2.255741	-1.95945	21
46	7	-0.026111	-0.02355		-0.019631		1
47	22		0.32760		0.328081	0.35231	1
48	5		-0.13876		-0.138761	0.25735	1
49	23		0.20178		0.275751	0.49912	1
50	7	-0.241491	-0.23324		-0.206131		10
51	22		0.10292	0.10840	0.110911		10
52	5		-1.97558		-1.975561	18.64163	10
53	23		0.27362	0.38975	0.554801		10
54	7		0.82410		0.927611	1.01804	0
55	22		21.73051		21.730531	50.41636	0
56	5		28.22050		28.220511	65.64801	0
57	23		0.25994		0.259991	0.27806	0
58	20		-0.50071		-0.500711	-0.39168	2
59	20		-0.05049		-0.050491	-0.03389	3
60	4	-0.016751	0.01297		0.012981		20
61	2		0.51116	0.54625	0.548301		1
62	3		0.29632		0.296561	0.29966	1
63	4		0.30130		0.301301	0.30437	1
64	10		-0.16497		-0.163401	-0.15692	2
65	10		0.24728		0.247331	0.29545	3
66	4		0.38866		0.499491	0.52925	10
67	2		0.15425		0.221601	0.23032	0
68	3		0.12111	0.22635	0.258281		0
69	4		0.19506	0.19842	0.199411		0
70	1	-0.246251	-0.17130		-0.171301		21
71	20	-0.080351	-0.07224		-0.042461		21
72	20		0.27950		0.279501	0.34179	1
73	10		-0.43299	-0.32376	-0.242241		21
74	10		-0.41106	-0.25484	-0.180581		20

22	6	6	1	0	1	0.9	0.0	6.9	0.8	0.9
23	0	10	1	2	-7	0.6	0.8	1.1	-10.7	-7.0
27	4	5	0	4	3	0.0	1.0	6.1	-0.9	-0.1

POINT NUMBER	5									
440.0000	67.50000			916.8000		14.75000		2.560000		
1.321000	61.00300			0.7710000		10.30000		11.63000		
0.7300000	1.621000			0.1500000		1027.000		4.360000		
1.670000	0.0000000E+00			0.0000000E+00		0.0000000E+00		0.0000000E+00		
440	0	0	10*	0	3	0	13*	1.0	0.0	12.5*
		1	9	2	1	1	9	0.9	0.6	10.4
		2	7	3	2	1	7	0.3	0.0	8.4
		3	9	2	2	0	9	0.7	0.0	9.9
		4	6	6	1	0	1	0.0	0.0	6.0
		5	1	10	0	2	-7	0.0	0.4	2.4
		6	4	4	0	5	5	0.0	0.7	6.9
		7	9	3	0	1	7	0.0	0.3	9.7
		10	5	4	4	0	5	0.4	0.0	6.5
		20	2	8	1	2	-3	0.7	0.3	4.1
		21	1	9	1	2	-5	0.6	0.9	1.8
		22	4	8	0	1	-3	0.0	0.3	4.7
		23	1	8	1	3	-3	0.6	1.0	2.2
		27	4	5	0	4	3	0.0	0.8	5.5

POINT NUMBER	6									
441.0000	67.80000			920.3000		20.05000		2.050000		
1.321000	57.68000			0.9640000		12.04900		3.860000		
0.9250000	1.096000			9.0000004E-02		1269.000		6.888000		
2.640000	0.0000000E+00			0.0000000E+00		21.00000		21.00000		
441	21	21	10*	2	1	0	9	0.8	0.0	10.8
		0	9	0	4	0	13*	1.0	0.0	11.2*
		1	6	5	1	1	3	0.4	0.4	7.1
		2	6	5	2	0	3	0.9	0.0	7.4
		3	9	2	1	1	9	0.8	0.7	10.1
		4	3	9	1	0	-5	0.8	0.0	3.8
		5	1	11	0	1	-9	0.0	0.3	1.7
		6	3	9	0	1	-5	0.0	0.3	3.7
		7	5	7	1	0	-1	0.1	0.0	5.1
		10	2	5	6	0	3	0.4	0.0	3.1
		20	8	3	1	1	7	1.0	0.6	9.4
		22	5	6	1	1	1	0.6	0.3	6.3
		23	0	10	1	2	-7	0.9	0.7	1.4
		27	9	2	0	2	9	0.0	0.6	10.0

POINT NUMBER	7									
442.0000	68.00000			916.4000		18.12000		1.770000		
1.321000	62.37600			1.064000		12.17200		6.980000		
1.005000	1.225000			9.0000004E-02		1358.000		5.841000		
2.230000	0.0000000E+00			0.0000000E+00		1.000000		1.000000		
443	1	1	11*	0	2	0	13*	0.1	0.0	11.1
		0	9	0	4	0	13*	0.9	0.0	12.3*
		2	9	2	1	1	9	0.1	0.8	9.3
		3	9	3	1	0	7	0.4	0.0	9.4
		4	3	10	0	0	-7	0.0	0.0	3.0
		5	1	10	0	2	-7	0.0	1.0	1.8
		6	3	9	0	1	-5	0.0	0.2	3.8
		7	6	6	1	0	1	1.0	0.0	7.0
		10	4	4	5	0	5	0.4	0.0	5.6
		20	8	4	1	0	5	0.8	0.0	8.8
		21	3	8	1	1	-3	0.6	0.0	4.6
		22	4	8	1	0	-3	0.6	0.0	4.6
		23	0	10	1	2	-7	0.7	0.8	1.1
		27	8	4	0	1	5	0.0	0.2	8.8

POINT NUMBER	8									
444.0000	68.00000			917.0000		19.86000		2.370000		
1.321000	62.37600			1.064000		12.17200		6.980000		
1.005000	1.225000			9.0000004E-02		1358.000		5.841000		
2.230000	0.0000000E+00			0.0000000E+00		1.000000		1.000000		

POINT NUMBER	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1.321000	54.84500	0.8470000	11.64300	3.640000															
0.8170000	1.117000	0.1100000	1137.000	5.974000															
2.280000	0.0000000E+00	0.0000000E+00	21.00000	21.00000															
444 21	21 10*	1 2	0 11	0.7 0.0	11.1*	9.2*	8.7*												
	0 8	1 4	0 11	0.9 0.0	10.3	7.7	7.3												
	1 6	6 1	0 1	0.3 0.0	6.3	-0.5	2.0												
	2 8	3 1	1 7	0.2 0.1	9.0	5.1	3.3												
	3 7	4 1	1 5	1.0 0.3	8.7	4.5	2.7												
	4 3	7 2	1 -1	0.5 0.8	3.9	-5.3	-3.6												
	5 1	11 0	1 -9	0.0 0.4	1.6	-9.8	-2.9												
	6 2	8 0	3 -3	0.0 0.9	2.5	-6.0	-0.3												
	7 3	8 2	0 -3	0.5 0.0	3.9	-5.2	-5.3												
	10 4	0 9	0 13*	0.8 0.0	7.3	1.6	4.1												
	20 8	2 1	2 9	0.9 0.8	9.6	6.2	6.3												
	22 6	5 2	0 3	0.9 0.0	7.5	2.1	2.7												
	23 0	10 1	2 -7	0.8 0.7	1.5	-9.9	-6.3												
	27 4	4 0	5 5	0.0 0.9	6.7	0.3	0.3												

POINT NUMBER	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
445.0000	68.00000	914.1000	12.50000	2.860000															
1.321000	60.81800	0.6520000	9.323000	13.81000															
0.6150000	1.989000	0.1900000	876.0000	4.602000															
1.760000	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00															
445 0	0 11*	0 2	0 13*	0.9 0.0	12.8*	12.6*	9.2*												
	1 10	2 0	1 9	0.0 0.7	10.3	7.7	6.9												
	2 4	7 0	2 -1	0.0 0.7	5.2	-2.7	-1.3												
	3 8	2 1	2 9	0.1 0.5	9.5	6.0	6.2												
	4 7	4 1	1 5	0.1 0.3	7.9	2.6	3.5												
	5 3	9 0	1 -5	0.0 0.4	3.6	-5.9	-0.9												
	6 5	4 0	4 5	0.0 0.6	7.6	2.1	0.7												
	7 10	3 0	0 7	0.0 0.0	10.0	7.0	6.4												
	10 5	3 5	0 7	0.9 0.0	7.1	1.2	2.4												
	20 2	9 1	1 -5	0.7 0.7	3.0	-7.1	-6.5												
	21 1	10 1	1 -7	0.6 0.4	2.2	-2.3	-6.5												
	22 4	9 0	0 -5	0.0 0.0	4.0	-5.0	-3.3												
	23 1	7 1	4 -1	0.6 1.0	2.4	-8.2	-5.4												
	27 4	6 0	3 1	0.0 0.8	5.6	-1.8	-0.3												

POINT NUMBER	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1285.000	46.00000	912.5000	16.97000	2.240000														
1.321000	61.48300	0.9150000	7.850000	8.970000														
0.8700000	1.352000	0.1200000	1198.000	5.090000														
1.950000	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00														
1285 0	3 10*	2 1	0 9	0.4 0.0	10.4	7.8	7.1											
	0 8	0 5	0 13*	0.9 0.0	11.4*	9.9*	8.3*											
	1 6	5 2	0 3	0.6 0.0	6.8	0.6	0.7											
	2 8	2 3	0 9	0.9 0.0	9.7	6.4	6.0											
	4 4	7 1	1 -1	0.1 0.4	4.7	-3.6	-1.4											
	5 6	6 0	1 1	0.0 0.2	6.8	0.5	0.5											
	6 4	6 0	3 1	0.0 0.9	5.7	-1.6	0.0											
	7 9	3 1	0 7	0.2 0.0	9.2	5.4	5.5											
	10 4	3 6	0 7	0.8 0.0	7.2	1.5	1.2											
	20 5	6 1	1 1	0.5 0.1	6.4	-0.2	0.3											
	21 3	8 1	1 -3	0.5 0.8	3.7	-5.6	-3.9											
	22 6	5 2	0 3	0.5 0.0	6.6	0.2	2.1											
	23 1	10 1	1 -7	0.5 1.0	1.6	-9.9	-6.0											
	27 0	11 0	2 -9	0.0 0.6	0.8	-11.4	-1.9											

POINT NUMBER	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
1287.000	46.00000	912.3000	17.79000	2.020000													
1.321000	62.05500	1.002000	8.048000	7.710000													
0.9480000	1.265000	0.1000000	1290.000	5.480000													
2.100000	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00													
1287 0	0 9*	0 4	0 13*	0.8 0.0	11.5*	10.1*	8.5*										
	3 9*	2 1	1 9	0.4 0.9	9.5	5.9	5.6										
	1 6	5 2	0 3	0.6 0.0	6.8	0.6	0.7										

4	8	2	3	0	7	0.9	0.0	9.7	6.4	6.0
4	2	10	0	1	-7	0.0	0.8	2.2	-8.6	-5.7
5	7	5	0	1	3	0.0	0.5	7.5	2.1	1.3
6	3	6	0	4	1	0.0	0.9	5.4	-2.2	0.0
7	8	3	1	1	7	0.2	0.1	9.1	5.2	5.4
10	4	3	6	0	7	0.8	0.0	7.2	1.5	1.3
20	8	3	1	1	7	0.6	0.1	9.5	5.9	6.1
21	3	7	1	2	-1	0.5	0.9	4.1	-4.9	-3.2
22	5	6	2	0	1	0.4	0.0	5.5	-2.0	0.6
23	1	10	1	1	-7	0.6	1.0	1.6	-9.8	-6.0
27	0	11	0	2	-9	0.0	0.4	1.4	-10.3	-1.6

POINT NUMBER		12										
1289.000		46.00000		912.4000		17.82000		1.710000				
1.321000		63.71900		1.129000		7.744000		7.120000				
1.062000		1.236000		9.0000004E-02		1419.000		6.020000				
2.300000		0.0000000E+00		0.0000000E+00		0.0000000E+00		0.0000000E+00				
1289	0	0	10*	0	2	0	13*	0.9	0.0	12.3*	11.6*	9.0*
		1	7	3	2	1	7	0.8	1.0	7.9	2.9	2.7
		2	7	3	2	1	7	0.8	0.1	8.9	4.8	5.4
		3	8	3	1	1	7	0.2	0.7	8.5	4.0	4.1
		4	1	12	0	0	-11	0.0	0.0	1.0	-11.0	-7.3
		5	7	5	0	1	3	0.0	0.6	7.4	1.8	1.2
		6	4	6	0	3	1	0.0	0.7	5.1	-2.9	0.2
		7	9	3	1	0	7	0.9	0.0	9.9	6.7	6.2
		10	4	4	5	0	5	0.5	0.0	5.8	-1.3	-1.1
		20	6	4	1	2	5	0.6	0.4	8.2	3.5	3.8
		21	4	6	1	2	1	0.5	0.4	5.7	-1.6	-0.5
		22	5	7	1	0	-1	0.2	0.0	5.2	-2.7	0.3
		23	1	11	1	0	-9	0.6	0.0	1.6	-9.8	-6.0
		27	1	7	0	5	-1	0.0	1.0	2.6	-5.9	-0.5

POINT NUMBER		13										
1290.000		46.00000		912.7000		20.31000		2.200000				
1.321000		56.74000		0.9430000		8.251000		3.370000				
0.9080000		1.081000		0.1000000		1243.000		6.145000				
2.350000		0.0000000E+00		0.0000000E+00		21.00000		21.00000				
1290	21	21	11*	1	1	0	11*	0.6	0.0	11.6*	10.2*	9.3*
		0	8	1	4	0	11*	0.7	0.0	9.9	6.8	7.2
		1	5	7	1	0	-1	0.3	0.0	5.3	-2.5	0.0
		2	8	3	1	1	7	0.6	0.2	9.4	5.7	4.0
		3	7	4	1	1	5	0.9	0.1	8.8	4.6	3.0
		4	1	10	1	1	-7	0.2	0.9	1.3	-10.4	-7.1
		5	6	7	0	0	-1	0.0	0.0	6.0	-1.0	0.5
		6	1	9	0	3	-5	0.0	0.7	2.8	-7.3	-0.2
		7	5	7	1	0	-1	0.6	0.0	5.6	-1.8	-2.1
		10	5	2	6	0	9	0.8	0.0	7.5	2.0	2.6
		20	8	2	1	2	9	0.7	0.7	9.8	6.6	6.9
		22	7	4	2	0	5	0.7	0.0	8.1	3.3	4.5
		23	1	10	1	1	-7	0.7	0.9	1.8	-9.4	-5.9
		27	2	8	0	3	-3	0.0	0.8	3.1	-6.8	-0.8

POINT NUMBER		14										
1292.000		46.00000		912.9000		20.17000		1.760000				
1.321000		59.90900		1.101000		8.463000		3.150000				
1.052000		1.068000		7.9999998E-02		1407.000		5.970000				
2.280000		0.0000000E+00		0.0000000E+00		21.00000		21.00000				
1292	21	20	10*	1	1	1	11*	0.6	0.0	11.6*	10.1*	9.7*
	21	21	9	3	1	0	7	0.5	0.0	9.5	6.0	6.4
		0	7	1	5	0	11*	0.9	0.0	9.3	5.7	5.5
		1	7	2	4	0	9	0.9	0.0	8.9	4.9	5.1
		2	6	3	4	0	7	0.9	0.0	8.2	3.3	2.4
		3	5	5	2	1	3	0.7	0.9	6.6	0.2	0.2
		4	1	12	0	0	-11	0.0	0.0	1.0	-11.0	-7.3
		5	6	7	0	0	-1	0.0	0.0	6.0	-1.0	0.5
		6	1	9	0	3	-5	0.0	1.0	2.1	-8.8	-0.5

7	4	6	2	1	1	0.1	0.8	4.4	-4.2	-2.2
10	5	2	6	0	9	0.6	0.0	7.9	2.9	2.6
22	5	4	2	2	5	0.9	0.4	7.7	2.4	3.6
23	2	8	1	2	-3	0.6	0.9	3.1	-6.8	-3.9
27	3	8	0	2	-3	0.0	0.1	4.7	-3.6	0.1

POINT NUMBER 15

1293.000	46.00000	912.9000	19.79000	2.290000
1.321000	49.72800	0.6910000	7.144000	1.230000
0.6710000	1.121000	0.1300000	949.0000	4.424000
1.690000	0.0000000E+00	0.0000000E+00	21.00000	21.00000
1293 21	21 10*	1 2	0 11	0.5 0.0 11.0 9.1 8.8*
21	22 10*	1 2	0 11	0.9 0.0 11.3* 9.7* 8.5
	0 7	4 2	0 5	0.5 0.0 7.9 2.9 4.4
	1 4	8 1	0 -3	0.6 0.0 4.6 -3.8 -2.6
	2 6	5 0	2 3	0.0 0.5 7.2 1.5 0.6
	3 4	7 0	2 -1	0.0 0.4 5.4 -2.3 -2.8
	4 4	7 0	2 -1	0.0 0.1 5.8 -1.2 1.0
	5 3	8 0	2 -3	0.0 0.5 4.4 -4.3 -0.3
	6 1	8 0	4 -3	0.0 0.8 3.1 -6.7 -0.2
	7 5	7 0	1 -1	0.0 0.1 5.9 -1.1 -0.9
	10 8	0 5	0 13*	0.5 0.0 10.0 7.0 6.3
	20 6	3 1	3 7	0.6 0.6 8.6 4.3 4.9
	23 3	2 1	7 9	0.6 0.9 4.9 -3.1 -1.5
	27 0	10 0	3 -7	0.0 1.0 0.7 -11.7 -2.0

POINT NUMBER 16

1294.000	46.00000	912.6000	20.13000	2.400000
1.321000	54.92700	0.8660000	8.036000	3.230000
0.8370000	1.095000	0.1100000	1158.000	5.641000
2.160000	0.0000000E+00	0.0000000E+00	21.00000	21.00000
1294 21	21 11*	1 1	0 11*	0.6 0.0 11.6* 10.1* 9.3*
	0 6	3 4	0 7	0.8 0.0 8.3 3.6 4.6
	1 5	7 1	0 -1	0.3 0.0 5.3 -2.4 0.0
	2 7	5 0	1 3	0.0 0.0 8.0 3.0 2.4
	3 6	5 0	2 3	0.0 1.0 6.9 0.9 -0.5
	4 2	7 2	2 -1	0.2 1.0 2.6 -7.9 -5.1
	5 4	7 0	2 -1	0.0 0.1 5.9 -1.2 1.2
	6 1	9 0	3 -5	0.0 0.6 2.8 -7.3 -0.3
	7 5	6 1	1 1	0.8 0.9 5.9 -1.2 -1.9
	10 6	1 6	0 11*	0.9 0.0 8.5 4.0 4.2
	20 8	2 1	2 9	0.6 0.6 9.9 6.7 7.1
	22 9	3 1	0 7	0.9 0.0 9.9 6.8 6.9
	23 2	8 1	2 -3	0.7 0.9 2.9 -7.3 -4.3
	27 1	9 0	3 -5	0.0 0.8 2.5 -8.0 -1.1

POINT NUMBER 17

1296.000	46.00000	912.3000	15.75000	2.500000
1.321000	61.26900	0.8390000	7.489000	10.59000
0.7960000	1.491000	0.1400000	1108.000	4.700000
1.800000	0.0000000E+00	0.0000000E+00	0.0000000E+00	0.0000000E+00
1296 0	7 9*	3 1	0 7	0.4 0.0 9.4 5.9 5.8
0	0 8	0 5	0 13*	0.8 0.0 11.5* 10.0* 8.2*
	1 8	3 2	0 7	0.7 0.0 8.8 4.7 4.3
	2 8	2 3	0 9	0.8 0.0 9.5 6.0 5.8
	3 7	2 1	3 9	0.3 0.4 9.6 6.1 5.7
	4 5	7 1	0 -1	0.2 0.0 5.2 -2.6 -0.8
	5 7	5 0	1 3	0.0 0.7 7.3 1.6 1.1
	6 4	4 0	5 5	0.0 1.0 6.3 -0.4 0.3
	10 4	3 6	0 7	0.7 0.0 7.3 1.6 1.1
	20 4	7 1	1 -1	0.5 0.2 5.3 -2.4 -1.6
	21 2	10 1	0 -7	0.4 0.0 2.4 -8.1 -5.6
	22 5	5 2	1 3	0.3 0.0 6.4 -0.1 0.6
	23 1	9 1	2 -5	0.5 1.0 1.6 -9.8 -5.9
	27 0	12 0	1 -11	0.0 0.8 0.2 -12.5 -2.1

POINT NUMBER		18		911.6000		13.94000		2.790000	
1298.000	0	46.00000	0.000000E+00	0	0	0.0	11.9*	10.8*	8.7*
1.321000	0	61.26300	0.000000E+00	1	0	7	0.6	0.0	6.0
0.6960000	0	1.741000	0.000000E+00	2	0	5	0.7	0.0	7.9
1.590000	0	0.000000E+00	0.000000E+00	3	3	7	0.6	0.9	6.5
	0	0	9*	4	0	13*	0.8	0.0	11.9*
	0	7	9*	1	0	7	0.6	0.0	9.6
	0	1	7	2	0	5	0.7	0.0	7.9
	0	2	4	3	3	7	0.6	0.9	6.5
	0	3	6	3	1	3	7	0.2	1.0
	0	4	7	4	0	2	5	0.0	0.3
	0	5	6	5	0	2	3	0.0	0.3
	0	6	8	2	0	3	9	0.0	0.7
	0	10	5	3	5	0	7	0.6	0.0
	0	20	2	8	1	2	-3	0.4	0.3
	0	21	2	10	1	0	-7	0.4	0.0
	0	22	4	7	1	1	-1	0.4	0.7
	0	23	2	8	1	2	-3	0.5	1.0
	0	27	0	11	0	2	-9	0.0	1.0

POINT NUMBER		19		912.3000		15.02000		2.640000	
1300.000	0	67.50000	0.000000E+00	0	0	13*	1.0	0.0	12.4*
1.321000	0	60.96100	0.000000E+00	1	0	9	0.8	0.0	10.3
0.7220000	0	1.583000	0.000000E+00	2	1	11	0.3	1.0	9.6
1.650000	0	0.000000E+00	0.000000E+00	3	1	11	0.3	1.0	9.6
	0	0	10*	3	0	13*	1.0	0.0	12.4*
	0	1	10*	1	0	9	0.8	0.0	10.3
	0	2	7	2	1	9	0.7	0.0	8.9
	0	3	9	1	2	11	0.3	1.0	9.6
	0	4	5	7	0	1	-1	0.0	0.6
	0	5	1	10	0	2	-7	0.0	0.4
	0	6	4	4	0	5	5	0.0	0.9
	0	7	7	3	1	2	7	0.7	0.3
	0	10	4	3	6	0	7	0.9	0.0
	0	20	2	8	1	2	-3	0.6	0.1
	0	21	1	9	1	2	-5	0.6	0.8
	0	22	4	6	1	2	1	0.1	0.4
	0	23	1	8	1	3	-3	0.6	1.0
	0	27	4	6	0	3	1	0.0	0.9

POINT NUMBER		20		912.4000		16.13000		2.410000	
1302.000	0	67.50000	0.000000E+00	0	0	9	0.0	0.0	11.0
1.321000	0	61.81300	0.000000E+00	1	0	13*	0.9	0.0	12.4*
0.8410000	0	1.444000	0.000000E+00	2	0	9	0.3	0.0	9.7
1.890000	0	0.000000E+00	0.000000E+00	3	0	9	0.3	0.0	9.7
	0	1	11*	2	0	9	0.3	0.0	9.7
	0	0	10	0	3	0	13*	0.9	0.0
	0	2	9	3	1	0	7	0.7	0.0
	0	3	9	2	2	0	9	0.3	0.0
	0	4	4	7	1	1	-1	0.1	0.3
	0	5	1	10	0	2	-7	0.0	0.4
	0	6	3	7	0	3	-1	0.0	0.7
	0	7	9	3	0	1	7	0.0	0.7
	0	10	4	4	5	0	5	0.5	0.0
	0	20	4	7	1	1	-1	0.7	0.0
	0	21	1	9	1	2	-5	0.6	0.6
	0	22	6	6	1	0	1	0.5	0.0
	0	23	0	10	1	2	-7	0.6	0.8
	0	27	4	5	0	4	3	0.0	1.0

POINT NUMBER		21		913.3000		17.11000		2.290000	
1304.000	0	67.50000	0.000000E+00	0	0	13*	0.8	0.0	12.5*
1.321000	0	62.44900	0.000000E+00	1	0	13*	0.8	0.0	12.5*
0.9200000	0	1.333000	0.000000E+00	2	0	13*	0.8	0.0	12.5*
2.040000	0	0.000000E+00	0.000000E+00	3	0	13*	0.8	0.0	12.5*
	0	0	11*	0	2	0	13*	0.8	0.0
	0	1	11*	0	2	0	13*	0.8	0.0
	0	2	11*	0	2	0	13*	0.8	0.0
	0	3	11*	0	2	0	13*	0.8	0.0
	0	4	11*	0	2	0	13*	0.8	0.0
	0	5	11*	0	2	0	13*	0.8	0.0
	0	6	11*	0	2	0	13*	0.8	0.0
	0	7	11*	0	2	0	13*	0.8	0.0
	0	8	11*	0	2	0	13*	0.8	0.0
	0	9	11*	0	2	0	13*	0.8	0.0
	0	10	11*	0	2	0	13*	0.8	0.0
	0	11	11*	0	2	0	13*	0.8	0.0
	0	12	11*	0	2	0	13*	0.8	0.0
	0	13	11*	0	2	0	13*	0.8	0.0
	0	14	11*	0	2	0	13*	0.8	0.0
	0	15	11*	0	2	0	13*	0.8	0.0
	0	16	11*	0	2	0	13*	0.8	0.0
	0	17	11*	0	2	0	13*	0.8	0.0
	0	18	11*	0	2	0	13*	0.8	0.0
	0	19	11*	0	2	0	13*	0.8	0.0
	0	20	11*	0	2	0	13*	0.8	0.0
	0	21	11*	0	2	0	13*	0.8	0.0
	0	22	11*	0	2	0	13*	0.8	0.0
	0	23	11*	0	2	0	13*	0.8	0.0
	0	24	11*	0	2	0	13*	0.8	0.0
	0	25	11*	0	2	0	13*	0.8	0.0
	0	26	11*	0	2	0	13*	0.8	0.0
	0	27	11*	0	2	0	13*	0.8	0.0

1	11*	2	0	0	9	0.0	0.0	11.0	9.0	8.1
2	10	3	0	0	7	0.0	0.0	10.0	7.0	7.1
3	8	2	1	2	9	0.3	0.3	9.9	6.8	7.1
4	4	8	0	1	-3	0.0	0.8	4.2	-4.6	-2.0
5	1	10	0	2	-7	0.0	0.7	2.1	-8.9	-2.5
6	3	7	0	3	-1	0.0	0.8	4.2	-4.6	-0.1
7	8	4	0	1	5	0.0	0.9	8.1	3.2	2.9
10	4	4	5	0	5	0.5	0.0	5.7	-1.6	-0.3
20	5	6	1	1	1	0.7	0.0	6.7	0.5	0.7
21	2	8	1	2	-3	0.6	0.3	4.1	-4.9	-3.8
22	4	8	1	0	-3	0.5	0.0	4.5	-3.9	-1.2
23	0	10	1	2	-7	0.6	0.8	1.1	-10.8	-7.0
27	5	4	0	4	5	0.0	1.0	6.9	0.8	0.2

POINT NUMBER	22									
1306.000	67.50000		912.4000		19.34000		1.720000			
1.321000	63.05300		1.138000		11.91600		6.140000			
1.073000	1.191000		9.000000E-02		1431.000		6.074000			
2.320000	0.0000000E+00		0.0000000E+00		1.000000		1.000000			
1306	1	0	10*	0	3	0	13*	1.0	0.0	12.5*
	1	1	10*	0	2	1	13*	0.0	0.0	11.0
	2	2	9	4	0	0	5	0.0	0.0	9.0
	3	3	9	3	1	0	7	0.3	0.0	9.3
	4	4	2	11	0	0	-9	0.0	0.0	2.0
	5	5	1	11	0	1	-9	0.0	0.2	1.8
	6	6	3	9	0	1	-5	0.0	0.2	3.8
	7	7	5	6	2	0	1	1.0	0.0	6.9
	10	10	4	3	6	0	7	1.0	0.0	6.2
	20	20	8	4	1	0	5	0.8	0.0	8.8
	21	21	5	6	1	1	1	0.6	0.7	5.9
	22	22	3	9	1	0	-5	0.0	0.0	3.0
	23	23	1	9	1	2	-5	0.7	0.8	2.1
	27	27	8	3	0	2	7	0.0	1.0	8.7

POINT NUMBER	23									
1307.000	67.40000		912.4000		20.09000		2.370000			
1.321000	56.76100		0.9370000		12.14000		3.740000			
0.9020000	1.096000		0.1100000		1236.000		6.600000			
2.520000	0.0000000E+00		0.0000000E+00		21.00000		21.00000			
1307	21	21	11*	1	1	0	11*	0.7	0.0	11.7*
	0	0	9	1	3	0	11*	1.0	0.0	11.0
	1	1	6	5	2	0	3	0.9	0.0	7.3
	2	2	8	4	1	0	5	0.7	0.0	8.7
	3	3	8	3	1	1	7	0.9	0.7	9.2
	4	4	3	9	1	0	-5	0.6	0.0	3.6
	5	5	1	10	0	2	-7	0.0	1.0	1.7
	6	6	2	9	0	2	-5	0.0	0.4	3.3
	7	7	4	8	1	0	-3	0.1	0.0	4.1
	10	10	2	4	7	0	5	0.4	0.0	3.3
	20	20	8	2	1	2	9	0.9	0.6	10.0
	22	22	6	5	1	1	3	0.7	1.0	6.7
	23	23	0	9	1	3	-5	0.8	1.0	1.4
	27	27	5	3	0	5	7	0.0	0.6	9.0

POINT NUMBER	24									
1309.000	67.50000		911.8000		17.09000		2.340000			
1.321000	62.57300		1.022000		11.58400		8.840000			
0.9630000	1.327000		0.1200000		1307.000		5.621000			
2.150000	0.0000000E+00		0.0000000E+00		0.0000000E+00		0.0000000E+00			
1309	0	0	11*	0	2	0	13*	0.8	0.0	12.6*
	0	1	11*	2	0	0	9	0.0	0.0	11.0
	2	2	9	4	0	0	5	0.0	0.0	9.0
	3	3	9	1	1	2	11	0.2	0.9	9.7
	4	4	3	9	0	1	-5	0.0	1.0	3.0
	5	5	1	10	0	2	-7	0.0	0.7	2.1
	6	6	3	7	0	3	-1	0.0	0.9	4.0

7	8	4	0	1	5	0.0	0.1	8.9	4.8	4.4
10	4	4	5	0	5	0.7	0.0	5.9	-1.3	0.1
20	7	5	1	0	3	0.8	0.0	7.8	2.5	2.6
21	2	8	1	2	-3	0.6	0.3	4.2	-4.6	-3.5
22	4	8	1	0	-3	0.3	0.0	4.3	-4.4	-1.4
23	0	10	1	2	-7	0.6	0.8	1.1	-10.9	-7.1
27	5	5	0	3	3	0.0	0.4	7.5	1.9	0.5

POINT NUMBER		25									
1310.000		67.40000		911.8000		17.93000		1.900000			
1.321000		62.73600		1.095000		11.85400		7.150000			
1.032000		1.234000		0.1000000		1385.000		6.034000			
2.310000		0.0000000E+00		0.0000000E+00		1.000000		1.000000			
1310	1	0	11*	0	2	0	13*	0.8	0.0	12.5*	12.1*
	1	1	11*	2	0	0	9	0.0	0.0	11.0	9.0
		2	8	5	0	0	3	0.0	0.0	8.0	3.0
		3	9	3	1	0	7	0.3	0.0	9.3	5.6
		4	2	11	0	0	-9	0.0	0.0	2.0	-9.0
		5	1	10	0	2	-7	0.0	1.0	1.8	-9.3
		6	3	9	0	1	-5	0.0	0.2	3.8	-5.4
		7	8	5	0	0	3	0.0	0.0	8.0	3.0
		10	4	4	5	0	5	0.9	0.0	6.0	-1.0
		20	8	4	1	0	5	0.8	0.0	8.8	4.6
		21	4	7	1	1	-1	0.6	0.0	5.6	-1.8
		22	4	8	1	0	-3	0.1	0.0	4.1	-4.8
		23	0	10	1	2	-7	0.7	0.8	1.1	-10.9
		27	9	4	0	0	5	0.0	0.0	9.0	5.0

POINT NUMBER		26									
1311.000		67.30000		911.6000		19.82000		2.650000			
1.321000		54.74900		0.8290000		11.58500		2.540000			
0.8010000		1.120000		0.1200000		1113.000		5.752000			
2.200000		0.0000000E+00		0.0000000E+00		21.00000		21.00000			
1311	21	21	10*	1	2	0	11	0.7	0.0	11.2*	9.4*
		0	8	2	3	0	9	1.0	0.0	10.4	7.8
		1	6	5	2	0	3	0.1	0.0	6.1	-0.8
		2	9	4	0	0	5	0.0	0.0	9.0	5.0
		3	9	4	0	0	5	0.0	0.0	9.0	5.0
		4	3	8	1	1	-3	0.4	0.7	3.7	-5.6
		5	1	11	0	1	-9	0.0	0.4	1.6	-9.8
		6	2	8	0	3	-3	0.0	0.9	3.2	-6.5
		7	3	8	1	1	-3	0.5	0.9	2.6	-5.9
		10	6	0	7	0	13*	0.9	0.0	8.4	3.8
		20	8	2	1	2	9	0.9	0.5	10.1	7.1
		22	6	5	2	0	3	0.9	0.0	7.4	1.8
		23	0	10	1	2	-7	0.8	0.7	1.5	-10.0
		27	3	6	0	4	1	0.0	0.7	5.9	-1.3

PLACEMENT OF IDENTIFIED SETS

	YVOTE	CVOTE	WVOTE	BVOTE	PVOTE
UNIQUE 1ST PLACE	11	12	19	19	20
TIED 1ST PLACE	9	6	0	0	0
1 WITH > VOTES	4	6	6	6	5
2 WITH > VOTES	2	0	0	0	0
3 WITH > VOTES	0	1	1	1	1
4 WITH > VOTES	0	1	0	0	0
5 WITH > VOTES	0	0	0	0	0
6 WITH > VOTES	0	0	0	0	0
7 WITH > VOTES	0	0	0	0	0
8 WITH > VOTES	0	0	0	0	0
9 WITH > VOTES	0	0	0	0	0
10 WITH > VOTES	0	0	0	0	0
11 WITH > VOTES	0	0	0	0	0
12 WITH > VOTES	0	0	0	0	0
13 WITH > VOTES	0	0	0	0	0

DISCRIMINATION SUCCESS BY SET, USING B VOTES														
SETS >	SET ID													
VOTES	0	1	2	3	4	5	6	7	10	20	21	22	23	27
0	12	1	0	0	0	0	0	0	0	0	6	0	0	0
1	0	3	0	0	0	0	0	0	0	0	3	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TOTAL	12	4	1	0	0	0	0	0	0	0	9	0	0	0

DISCRIMINATION SUCCESS BY SET, USING P VOTES														
SETS >	SET ID													
VOTES	0	1	2	3	4	5	6	7	10	20	21	22	23	27
0	12	1	0	0	0	0	0	0	0	0	7	0	0	0
1	0	3	0	0	0	0	0	0	0	0	2	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
TOTAL	12	4	1	0	0	0	0	0	0	0	9	0	0	0

DISCRIMINATION SUCCESS USING B & P VOTES JOINTLY

P & B AGREE, ON IDENTIFIED SET	19
B NOT P SELECTS IDENTIFIED SET	0
P NOT B SELECTS IDENTIFIED SET	1
P & B DISAGREE, ON OTHER SETS	1
B & P AGREE, ON ONE OTHER SET	5
TOTAL	26
B & P VOTE RANK OF IDENTIFIED SET	
BRANK	1= 2= OTHER= PRANK
1	19 0 0
2	1 5 0
OTHER	0 0 1

SUCCESS OF SEVERAL CHOICE RULES

CHOOSE SET WITH B1=P1, ELSE NO CHOICE

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB
CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	SUCCESS
0	12	3	0	0	0	0	0	0	0	0	1	0	0	0	0.636
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000

21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 24 TRIALS; 0.71 TO 0.86 PROB SUCCESS

CHOOSE SET WITH B1

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	1	0	0	0	0	0	0	0	1	0	0	0	0.595
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 26 TRIALS; 0.65 TO 0.80 PROB SUCCESS

CHOOSE SET WITH P1

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	0	0	0	0	0	0	0	0	1	0	0	0	0.634
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.820
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

20 SUCCESSES IN 26 TRIALS; 0.69 TO 0.84 PROB SUCCESS

CHOOSE SETS WITH (B1 OR B2)&(P1 OR P2)

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0.891
1	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0.707
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0.728
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

24 SUCCESSES IN 26 TRIALS; 0.95 TO 0.96 PROB SUCCESS

CHOOSE SETS WITH B1 OR B2 OR P1 OR P2

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
--------	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----------------

0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0.891
1	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0.707
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0.357
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

25 SUCCESSES IN 24 TRIALS: 0.90 TO 0.99 PROB SUCCESS

CHOOSE SET WITH B1=P1, ELSE WITH B2=P2, ELSE NO CHOICE

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	0	0	0	0	0	0	0	0	1	0	0	0	0.134
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 24 TRIALS: 0.71 TO 0.96 PROB SUCCESS

CHOOSE SET WITH B1=P1, ELSE B2=P2, ELSE B2=P1, ELSE B1=P2, ELSE NO CHOICE

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	0	0	0	0	0	0	0	0	1	0	0	0	0.424
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 24 TRIALS: 0.65 TO 0.90 PROB SUCCESS

CHOOSE SET WITH B1=P1, ELSE B2=P1, ELSE B1=P2, ELSE B2=P2, ELSE P1

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	0	0	0	0	0	0	0	0	1	0	0	0	0.424
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

20	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 26 TRIALS; 0.65 TO 0.80 PROB SUCCESS

CHOOSE SET WITH B1=P1, ELSE B1=P2, ELSE B2=P1, ELSE B2=P2, ELSE P1

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	0	0	0	0	0	0	0	0	1	0	0	0	0.636
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 26 TRIALS; 0.65 TO 0.80 PROB SUCCESS

CHOOSE SET WITH B1=P1, ELSE B1=P2, ELSE NO CHOICE

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	0	0	0	0	0	0	0	0	1	0	0	0	0.636
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 25 TRIALS; 0.68 TO 0.83 PROB SUCCESS

CHOOSE SET WITH B1=P1, ELSE B1=P2, ELSE B1

CHOICE	0	1	2	3	4	5	6	7	10	20	21	22	23	27	L PROB SUCCESS
0	12	3	1	0	0	0	0	0	0	0	1	0	0	0	0.595
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0.250
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
21	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0.794
22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.000
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000

19 SUCCESSES IN 26 TRIALS; 0.65 TO 0.80 PROB SUCCESS

DISCRIMINATION SUCCESS BY HYPERPLANE
 HYPERPLANE - POINTS IN THE FIRST SET POINTS IN THE SECOND SET

PLANE#	SET#1	TVOTE	NVOTE	UVOTE	GVOTE	Y%	TVOTE	NVOTE	UVOTE	GVOTE	Y%
1	0	6	12	0	0	100.					
2	0	27	12	0	0	100.					
6	1	6	4	0	0	100.					
7	1	27	3	0	0	75.					
8	21	6	9	0	0	100.					
9	21	27	9	0	0	100.					
26	2	3	0	1	0	0.					
27	2	4	1	0	0	100.					
28	2	6	1	0	0	100.					
29	2	27	1	0	0	100.					
35	21	2	9	0	0	100.	1	0	0	0	100.
36	21	3	9	1	0	89.					
37	21	4	9	0	0	100.					
42	21	7	9	0	0	100.					
43	21	22	5	1	3	56.					
44	21	5	9	0	0	100.					
45	21	23	9	0	0	100.					
46	1	7	0	1	3	0.					
47	1	22	4	0	0	100.					
48	1	5	4	0	0	100.					
49	1	23	4	0	0	100.					
54	0	7	12	0	0	100.					
55	0	22	12	0	0	100.					
56	0	5	12	0	0	100.					
57	0	23	12	0	0	100.					
58	2	20	1	0	0	100.					
61	1	2	3	0	1	75.	0	0	1	0	0.
62	1	3	4	0	0	100.					
63	1	4	4	0	0	100.					
64	2	10	1	0	0	100.					
67	0	2	7	0	5	58.	0	0	1	0	0.
68	0	3	0	0	12	0.					
69	0	4	7	0	5	58.					
70	21	1	8	1	0	89.	4	0	0	0	100.
71	21	20	8	1	0	89.					
72	1	20	4	0	0	100.					
73	21	10	0	0	9	0.					
75	0	20	12	0	0	100.					
76	21	0	0	9	0	0.	12	0	0	0	100.
77	1	10	4	0	0	100.					
78	1	0	0	1	3	0.	7	0	5	0	58.
79	0	10	0	0	12	0.					
80	2	7	0	1	0	0.					
81	2	22	1	0	0	100.					
82	2	5	1	0	0	100.					
83	2	23	1	0	0	100.					

PLANE#	SET#1	RELINDEX	SET #2	RELINDEX
1	0	0.9646919	6	0.0000000E+00
2	0	0.9701428	27	0.0000000E+00
3	6	0.0000000E+00	27	0.2500000
4	10	0.9521406	6	0.0000000E+00
5	10	0.9632926	27	0.2500000
6	1	0.9804674	6	0.0000000E+00
7	1	0.9139677	27	0.2500000
8	21	0.7885437	6	0.0000000E+00
9	21	0.9597888	27	0.0000000E+00
10	20	0.9635404	6	0.2500000
11	20	0.9635404	27	0.0000000E+00
12	7	0.9154809	22	0.8988510
13	7	0.9154809	5	0.5000001
14	7	0.9154809	6	0.2500000
15	7	0.9154809	27	0.0000000E+00
16	7	0.9562657	23	0.4563599
17	22	0.8988510	5	0.1339746
18	22	0.8988510		

18	21	0.8988510	6	0.0000000E+00
19	22	0.8988510	27	0.2500000
20	22	0.8988510	23	0.7071068
21	5	0.5000001	6	0.0000000E+00
22	5	0.5000001	27	0.2500000
23	5	0.1339746	23	0.2429810
24	6	0.2500000	23	0.7071068
25	27	0.2500000	23	0.7071068
26	2	0.7738953	2	0.5873718
27	2	0.9526393	4	0.9117225
28	2	0.9526393	6	0.0000000E+00
29	2	0.9526393	27	0.2500000
30	3	0.9822294	4	0.9117225
31	3	0.9822294	6	0.2500000
32	3	0.9389312	27	0.2500000
33	4	0.6830750	6	0.0000000E+00
34	4	0.7550659	27	0.2500000
35	21	0.8722839	2	0.9311870
36	21	0.9152243	3	0.8822294
37	21	0.8979173	4	0.6830750
38	20	0.9848267	7	0.9774109
39	20	0.9811892	23	0.7201843
40	20	0.9635404	5	0.1339746
41	20	0.9635404	23	0.7071068
42	21	0.9239616	7	0.9542457
43	21	0.5773315	22	0.1991577
44	21	0.9787731	5	0.5000001
45	21	0.9597822	23	0.6299605
46	1	0.6474457	7	0.2550944
47	1	0.9139677	22	0.8988510
48	1	0.9139677	5	0.1339746
49	1	0.9804674	23	0.7071068
50	10	0.8456879	7	0.5685425
51	10	0.7682495	22	0.3368225
52	10	0.9195263	5	0.0000000E+00
53	10	0.0000000E+00	23	0.0000000E+00
54	0	0.4862671	7	0.0000000E+00
55	0	0.9646919	22	0.0000000E+00
56	0	0.9646919	5	0.0000000E+00
57	0	0.9813281	23	0.7071068
58	2	0.9526393	20	0.8848267
59	3	0.8294983	20	0.9469997
60	20	0.9469997	4	0.9117225
61	1	0.1513519	2	0.3244324
62	1	0.8358765	3	0.5414429
63	1	0.9381139	4	0.9303528
64	2	0.7362671	10	0.9632926
65	3	0.7787781	10	0.9576904
66	10	0.5060577	4	0.0000000E+00
67	0	0.4528961	2	0.2720865
68	0	0.1124115	2	0.0000000E+00
69	0	0.8165894	4	0.0000000E+00
70	21	0.8893280	1	0.9300045
71	21	0.5853424	20	0.5841370
72	1	0.9358765	20	0.8348267
73	21	9.0530396E-02	10	4.7832308E-03
74	20	0.1068039	10	1.4301059E-03
75	0	0.9122938	20	0.1498456
76	21	4.0488910E-02	0	0.9381568
77	1	0.4277802	10	0.7785034
78	1	3.0886175E-02	0	0.6042927
79	0	4.6000811E-03	10	1.4301059E-03
80	2	0.8703309	7	0.9154809
81	2	0.4112701	22	0.4089946
82	2	0.9526393	5	0.5000001
83	2	0.9755488	23	0.7071068

END

DATE

FILMED

9-88

DTIC